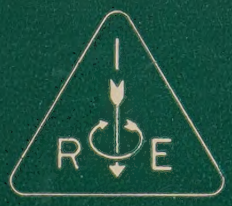
IRE
Transactions
on ELECTRONIC COMPUTERS



Volume EC-7 DECEMBER, 1958 Number 4

Published Quarterly

TABLE OF CONTENTS

CONTRIBUTIONS

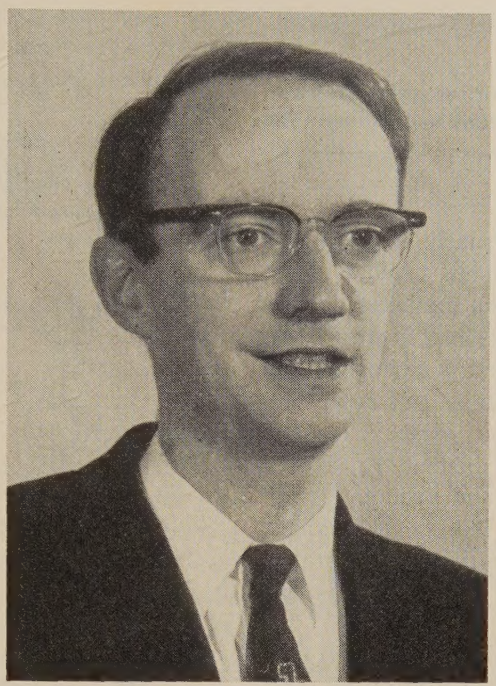
CORRESPONDENCE

PUBLISHED BY THE
Professional Group on ELECTRONIC COMPUTERS

IRE Transactions® on Electronic Computers

Published by the Institute of Radio Engineers, Inc., for the Professional Group on Electronic Computers at 1 East 79th Street, New York 21, N.Y. Responsibility for the contents rests upon the authors and not upon the IRE, the Group, or its members. Price per copy: IRE-PGEC members, $1.00; IRE members, $1.50, nonmembers, $3.00. Yearly subscriptions rate: nonmembers, $17.00; colleges and public libraries, $12.75. Address requests to The Institute of Radio Engineers, 1 East 79th Street, N.Y. 21, N.Y.

Notice to Authors: Address all papers and editorial correspondence to J. P. Nash, Missile Systems Division, Lockheed Aircraft Corp., 3251 Hanover Street, Palo Alto, Calif. To avoid delay, 3 copies of papers and figures should be submitted, together with the originals of the figures which will be returned on request. All material will be returned if a paper is not accepted.



Howard E. Tompkins

Beginning with the June, 1959, issue, IRE TRANSACTIONS ON ELECTRONIC COMPUTERS will have a new editor. He is Howard E. Tompkins, assistant professor of electrical engineering in The Moore School of Electrical Engineering at the University of Pennsylvania, Philadelphia, Pa.

Dr. Tompkins is especially well qualified for the post. He was one of the early workers in the digital computer field, and continues to be active in it. His education has been directed toward computer applications of electrical engineering. He received the B.A. degree in physics from Swarthmore College, Swarthmore, Pa., in 1942.

From 1942 to 1947, he was employed by the Philco Corporation in their Research Division, working first on microwave resonators, receivers, power meters, and measuring techniques, then on magnetic recording techniques and phonograph pickups. Several patents resulted.

In 1947, he received the M.S. degree in physics from the University of Pennsylvania, and that fall, joined the staff of the University as Instructor in The Moore School of Electical Engineering. Until 1951, he taught undergraduate electrical engineering, took graduate courses, and worked, first on electromedical instrument amplifiers, then on digital computer components.

In 1951, he joined the Burroughs Corporation as head of a transistor circuits research group, working on transistor switching circuits for computers, including in particular transistor-resistor logic. In 1955, he became Educational Services Manager at the Burroughs Research Center, returning to The Moore School in the fall of 1956.

During 1956–57, he supervised graduate student research on transistor high-frequency measurements, and wrote his dissertation on Unit-distance Codes. He received the Ph.D. degree in electrical engineering in 1957 from the University of Pennsylvania.

Over the years he has been active in local IRE affairs. He was local Arrangements Chairman for the first IRE-AIEE-ACM Computer Conference, held in Philadelphia in December, 1951. For many years he has been associated with the IRE-AIEE Transistor and Solid-State Circuits Conference, either as a member or chairman of the Program Committee, as Publicity Chairman, or as IRE Philadelphia Section representative. He was chairman of the Philadelphia Chapter of IRE-PGCT for 1956–57. He was IRE-PGEC representative to the Program Committee of the 1955 IRE National Convention, at which he was organizer and moderator of the Symposium, "Design of Machines to Simulate the Behavior of the Human Brain." (See IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-5, pp. 240–255; December, 1956.) Dr. Tompkins is also a member of AIEE, ACM and ASEE.

The two and one-half years during which I have been editor have been pleasant and stimulating ones. The interest in computers has caused a growth of our group that has brought the number of copies of these TRANSACTIONS published each quarter from 6300 to 8700, and there are positive indications that this is only the beginning.

Under Dr. Tompkins' editorship, we shall see these TRANSACTIONS continue its growth, and he will need the assistance of many people in carrying out his responsibilities. I know that those who have been helping me, especially in the arduous task of reviewing papers, will continue to give their support to Dr. Tompkins. I want to express my great appreciation to those who have given so generously of their time, and who must do so without public acknowledgment in order that our reviewing procedures may have the anonymity required.—J. P. NASH.

A Magnetic Core Parallel Adder*

MAO-CHAO CHEN†

Summary—A logical design using magnetic core elements which does not have the usual carry time limitations is described. The synthesis uses a truth-table technique.

Introduction

BINARY computer units are usually analyzed by Boolean algebra, then built from "And," "Or," and "Not" switches by various methods. In the present paper, a truth-table type synthesis is employed. The addition function to be described is performed by the voltage (flux) addition, subtraction and switching characteristics of magnetic memory cores. Previous work in this field has been done by Stifler [1], Mealy [2], more recently by Andrew [3], and others. An extension of these early methods has led to an arrangement for the addition function resulting in a parallel adder that does not have the usual carry-time limitation.

The voltage addition and subtraction operation employed makes admissible both negative numbers and numbers greater than 1. For example in Fig. 1 the value at point S_2 (to be described later) is 2 when $c=x=1$, and $y=0$; the value c' in (4) is -1, when x, y, and c are zero. Numbers in Boolean algebra are restricted to 1 and 0, and the logical operations "And," "Or," and "Not" are employed to mechanize the algebra in the binary system. Furthermore, Boolean functions do not generally describe voltage addition and subtraction. In consequence, Boolean algebra cannot completely describe the behavior of a multilevel system, even though the resulting numbers (1 and 0) are the same. Therefore, in the following description, ordinary algebra and the usual meaning of addition and subtraction are employed.

The Adder

One stage of the adder is shown in Fig. 1. Let x and y denote the addend and augend respectively, c the input carry, and s and c' the sum and output carry. All magnetic cores are identical, and rectangular magnetization curves are assumed. Each driving winding of the cores A, B, C, and D, has the same number of N turns. Each output or secondary winding also has same number of turns. A "1" on x or y is represented by a current pulse of magnitude I amps (flowing in the direction shown) providing a net amp-turn (NI) is large enough to drive each of the cores into saturation. An "0" for x or y provides no current at pulse time. The x- and y-pulses are applied at the same instant of time.

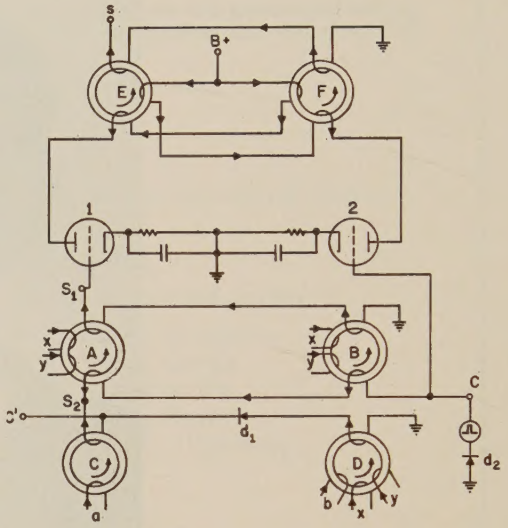


Fig. 1—One stage of adder. The arrows on input windings x, y and auxiliary windings a, b show current directions. Arrows on output windings show directions of induced emf's when the cores change flux. The concentric arc arrow ⌣ inside the two circles shows flux direction before operation. All x windings and y windings are in series; they are shown separately for clearness.

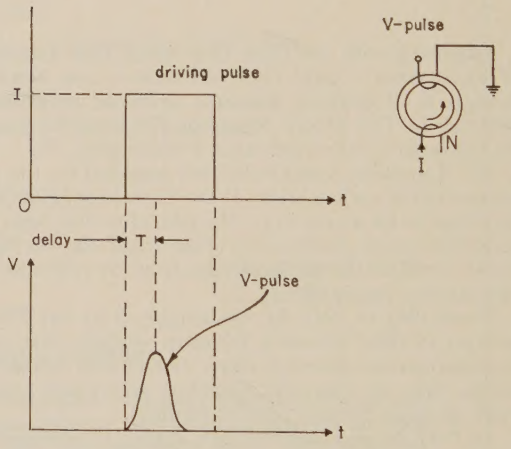


Fig. 2—Pulse obtained when a core is switched by NI amp-turns.

In these circuits, we are dealing mainly with pulses as shown in Fig. 2. For convenience we define a V-pulse as one which is identical with that shown in Fig. 2 not only in shape and magnitude but in time as well. We initially assume that the Thevenin's equivalent circuit at the input carry c, looking towards lower bits, is a pulse generator and a diode d_2 in series as shown in Fig. 1. The pulse is a V-pulse when c is "1," and zero when c is "0." A current pulse I is applied to the windings b of core D and a of core C at the same instant of time when

* Manuscript received by the PGEC, November 27, 1956; revised manuscript received, March 4, 1958.

† Physics Dept., Stanford University, Stanford, Calif.

x- and y-pulses are applied. A driving mmf is considered negative when it tends to maintain flux in its original direction, and positive when in the opposite direction. Thus a core changes state when the net driving mmf is greater than NI, and remains unchanged when it is zero or negative. Table I gives net mmf's of the cores A, B, C, and D for various combinations of x and y.

TABLE I
NET MAGNETOMOTIVE FORCE AS A FUNCTION OF x AND y

	core A	core B	core C	core D
$x=0, y=0$	0	0	NI	$-NI$
$x=0, y=1$	$-NI$	NI	NI	0
$x=1, y=0$	NI	$-NI$	NI	0
$x=1, y=1$	0	0	NI	NI

Let us define "1" as the condition when an output winding on a core is a V-pulse and "0" when there is no pulse. From Table I, we see that core A changes flux only when $x=1$ and $y=0$. Hence, the output of core A is $x(1-y)$. Similarly, the output of core B is $y(1-x)$. Since the output windings of A and B are in series, the value at S_1 is

$$S_1 = x(1 - y) + y(1 - x) = x + y - 2xy. \quad (1)$$

S_1 and input carry c are applied to the grids of two vacuum tubes, the plate circuits of which are connected through two cores E and F as shown in Fig. 1. Normally the dc plate currents flow through either of the cores in opposite directions, and their mmf's cancel each other. When the S_1 pulse is applied alone, plate current of tube 1 increases causing the net mmf's in core E to become negative and core F positive. The amplifiers and driving winding turns are so adjusted that this change in mmf is large enough to switch core F. Similarly when input c-pulse is applied alone, core E changes flux. But when both S_1 and c are applied, their mmf's cancel each other again and neither core is switched. Hence core E changes flux only when c is 1 and S_1 is zero, thus it mechanizes the function, $c(1-S_1)$. Similarly core F mechanizes S_1 $(1-c)$. Since their output windings are in series, voltage at s is

$$s = S_1(1 - c) + c(1 - S_1) = c + S_1 - 2cS_1. \quad (2)$$

Substituting (1) for S_1, we have,

$$s = c + x + y - 2cx - 2cy - 2xy + 4cxy. \quad (2a)$$

This is the equation for the sum when c, x, and y are added, which can be easily checked by a truth table and therefore is not shown here.

From Table I, it can be seen that core C always provides an output at "pulse time" and is therefore a "1" generator. The value at S_2 is the same as S_1 except c is in series, thus $S_2 = S_1 + c$. The opposing direction of the output winding on core C, is selected so that the value at c' is

$$c' = S_2 - 1 = S_1 + c - 1. \quad (3)$$

Substituting (1) for S_1 results in,

$$c' = x + y - 2xy + c - 1. \quad (4)$$

When x, y, and c are zero, c' becomes -1. Due to diode d_2, no negative voltage can appear at c', so

$$c' = f(x + y - 2xy + c - 1), \quad (5)$$

where the function f takes the value of its argument when it is positive, and zero when it is negative. From Table I, it is seen that core D mechanizes the logical "And" (x_y).

$$c' = xy. \quad (6)$$

Thus c' consists of two branches in parallel, one from core C to A, B, carry c, and diode d_2 to ground, the other from diode d_1, core D to ground. These two branches could form a closed loop. Hence, diode d_1, is inserted to prevent circulating current within this loop. Hence c' then provides the voltage of either of the branches and the greater is then the output. From (5) and (6) the complete expression for the output carry is

$$c' = \begin{cases} f(c+x+y-2xy-1), & \text{if } f(c+x+y-2xy-1) > xy \\ xy, & \text{if } xy > f(c+x+y-2xy-1) \end{cases} \quad (7)$$

This expression satisfies all of the cases when x, y, and c are added, and can be checked easily by a truth table.

Several important points are to be noted in the above circuit:

1) None of the logic cores or the equivalent input carry generator circuit is loaded.

2) All of the cores (A, B, C, and D) are switched by the same number of amp-turns (NI) and at the same instant of time.

3) The Thevenin equivalent circuit from c' toward lower bits is the same as that at c, *i.e.*, a pulse generator and a diode in series.

The results 1) and 2) lead to the fact that all voltage pulses induced have the same shape, magnitude, and time instant as shown in Fig. 2, and are all V-pulses. This is essential so that the quantities in (7) can be manipulated arithmetically. Therefore the output carry c' is also a V-pulse. Since the input carry c and output carry c' are identical in their pulse representation as well as their equivalent circuit, all stages can be connected in cascade, and a complete adder is formed. The two lowest stages are drawn in Fig. 3, the extension to many stages being obvious. Since there is no input carry in the first stage, the triode 1 and core F are primarily unnecessary. They are inserted there so that the sum pulse can appear at the same time as other sum pulses.

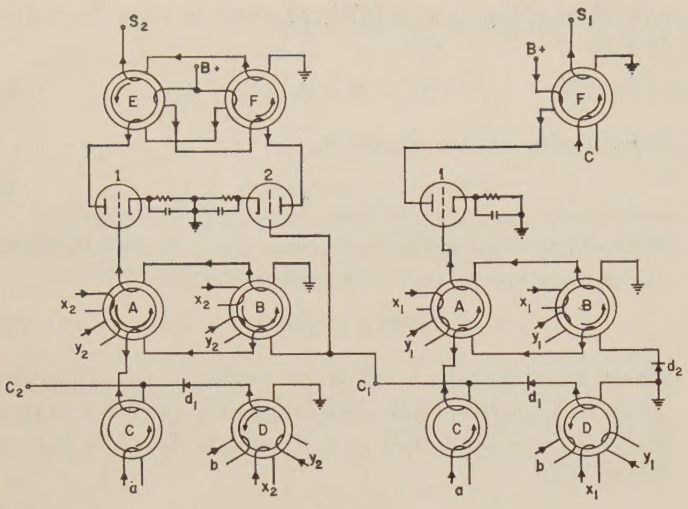


Fig. 3—First two stages of adder.

Discussion

As shown in Fig. 2, there is always a time delay between the driving pulse and the output V-pulse. A review of the operation of this adder shows that this kind of delay does not effect its operation. The only effect is in the operation speed. In performing the addition, first, cores A, B, C, and D, are switched, then the output pulses of these cores are used to switch cores E and F. Let T denote the time delay in switching a single core. The speed of operation is $2T$.

It is a well-known switching characteristic that secondary output pulses will be changed in shape, magnitude, and pulse width if secondary windings supply current. As stated earlier, no core in this adder is loaded. This is essential in order to provide uniform pulse shapes.

Vacuum tube amplifiers are used to amplify and convert voltage pulses into current pulses to drive cores E and F. Transistor amplifiers draw current as well as power from the signal; thus the V-pulses would be distorted if they were used instead of vacuum tube amplifiers. For this reason, vaccuum tube amplifiers are preferable.

Under no-load condition, the output voltage obtained in switching a core might be quite large depending upon driving amp-turns. For instance, 2.5 volts can be obtained across a single output turn when 2 amp-turns is used to drive a General Ceramics F-483 ferrite core. When a high transconductance tube is used, for instance, a 6AG7, $g=7700$ μmho, this voltage can be transformed into a current pulse of $2.5\times7700\times10^{-6}=0.019$ amp. Thus 100 turns winding is sufficient to drive cores E or F. It is obvious that high primary impedances do not effect the operation of these cores. If in application a large pulse across s is not necessary, cores of small sizes can be used for cores E and F, and driving amp-turns can be greatly reduced.

In Fig. 3, we see that a carry pulse propagates through windings of cores B, A, and C, of successive stages until it arrives at a stage where both x and y are zero. Then it transforms into a sum pulse and disappears from the circuit. But, in general, it can propagate as far as to the highest bit. Due to impedances of output windings of these cores, the carry pulse is delayed and distorted during propagation, and the initial assumption that a carry pulse is a V-pulse can not be justified. This puts on the restriction that the number of bits to be added is limited when the effect of pulse delay and distortion is beyond tolerance. However, it is possible to reduce the delay effect by making use of what has already been shown; one turn secondary winding is enough to drive the amplifier. This leads one to arrange the A, B, C cores of successive stages coaxially, a center conductor serving the secondary windings of cores A, B, C. This looks like a concentric transmission system with cores as dielectric and carry pulse propagating along the center conductor. It is obvious that time delay can be further reduced by using cores of less thickness. In order to keep a definite amount of flux to give sufficient voltage, core diameter must be increased accordingly. Hence the maximum allowable number of bits to be added depends upon particular design and is best determined experimentally.

Reset windings must be provided for all cores to set the cores to their initial conditions before the operation of the next addition.

Bibliography

[1] W. W. Stifler, ed. "High-Speed Computing Devices," Eng. Res. Assoc. Staff, McGraw-Hill Book Co. Inc., New York, N. Y.; 1950.
[2] G. H. Mealy, "A method for synthesizing sequential circuits," *Bell Sys. Tech. J.*, vol. 34, pp. 1045–1079; September, 1955.
[3] L. Andrews, "A technique for using memory cores as logical elements," *Proc. EJCC*, pp. 39–46; December, 1956.
[4] R. D. Kodis, "Application and performance of magnetic core circuits in computing systems," *Proc. EJCC*, pp. 30–34; December, 1954.
[5] E. A. Sands, "Behavior of rectangular hysteresis loop magnetic materials under current pulse conditions," Proc. IRE, vol. 40, pp. 1246–1250; October, 1952.
[6] R. C. Minnick, "Magnetic switching systems," *J. Appl. Phys.*, vol. 25, pp. 479–485; April, 1954.
[7] A. Wang, "Magnetic triggers," Proc. IRE, vol. 38, pp. 626–629; June, 1950.
[8] J. R. Freeman, "Pulse responses of ferrite memory cores," *Proc. WESCON Computer Sessions*, pp. 50–61, August, 1954.
[9] M. Karnaugh, "Pulse switching circuits using magnetic cores," Proc. IRE, vol. 43, pp. 570–584; May, 1955.
[10] S. Gutterman, R. D. Kodis, and S. Ruhman, "Logical and control functions performed with magnetic cores," Proc. IRE, vol. 43, pp. 291–298; March, 1955.
[11] R. A. Ramey, "A single core magnetic amplifier as a computer element," *Proc. AIEE*, vol. 71, pp. 442–446; January, 1952.

Significant Digit Computer Arithmetic*

N. METROPOLIS† AND R. L. ASHENHURST†

Summary—**The usual floating point arithmetic makes error analysis difficult. This paper describes an alternative system which offers a means of analyzing floating point calculations more effectively and which also possesses certain advantages from an equipment standpoint.**

Introduction

MODERN computers which make use of so-called floating point arithmetic, where every number is represented as a fractional part with exponent, customarily deal with such numbers in normalized form; *i.e.*, nonzero fractional parts are required to lie between B^{-1} and 1 in magnitude, where B is the number system radix. The use of such a convention, however, leads to the retention of nonsignificant digits in computed results, which therefore gives them a highly deceptive appearance of accuracy with no indication of what their true precision might be. This paper describes some aspects of an alternative system, which may be termed *significant digit arithmetic.* First the rules for this arithmetic are described, and then methods of computer implementation of the rules are discussed. Although the basic philosophy underlying the significant digit system is indicated, a detailed justification is not attempted here. An analytic study of its error propagation properties is in preparation.

The following discussion applies quite generally to floating point arithmetic, irrespective of the number base and specific representation conventions used. For concreteness, however, it will be assumed that numbers are represented in binary form; the fractional part f consists of an n-digit fractional magnitude with sign appended, and the integral exponent e is also represented as a signed magnitude. Obvious modifications are necessary to make the discussion appropriate to the case of complement representation, biased exponents, etc.

Arithmetic Rules

A basic feature of floating point arithmetic is its property of automatic adjustment for overflow at the left. This feature is independent of the use of normalized form, however. The rules to be given apply to unnormalized operands, and have the property that the results give some indication of their expected precision. Besides this, the rules have certain advantages from the point of view of operation speed.

* Manuscript received by the PGEC, April 17, 1957; revised manuscript received, August 15, 1958. This work was supported by the U. S. Atomic Energy Commission.

† Institute for Computer Research, University of Chicago, Chicago, Ill.

Addition and subtraction

In performing floating point addition or subtraction, it is necessary that one or the other operand be shifted, and its exponent adjusted accordingly, so that the two exponents agree and the numbers are, so to speak, "lined up." In a normalized system, the number with the smaller exponent must be shifted right; otherwise digits would be lost at the left. It is conceivable that another procedure could be used for unnormalized operands for which some left shifting is allowed. However, it turns out that the desired significance properties are obtained if numbers are treated much as in a normalized system, except that no further adjustment is made at the completion of an addition or subtraction where there is no overflow. It should be noted that overflow may be expected to occur much less often when operands are not required to be in normalized form.

Multiplication

There is no fundamental necessity to shift operands in the case of multiplication; the product can be formed directly and the exponents added. If only the leftmost n digits of the resulting $2n$-digit product are retained, however, significant digits may be lost. Hence a readjusted product is defined, which has the property that the number of significant digits in the n-digit result is approximately the same as that for the factor with fewer significant digits. Conceptually, if that factor were multiplied by a value near 1, the desired effect would be automatically obtained. Hence the product can tentatively be defined as the result of multiplying the factor with the smaller fractional part by the other factor normalized, and adjusting the exponent accordingly. In this case, however, the high order part of the $2n$-digit fractional part product has either the same number of significant digits as the unnormalized factor or else one less significant digit. Thus, continued use of the rule in this form would introduce a tendency toward loss of significant digits. To counteract this, it is advisable to incorporate an additional adjustment of the product, consisting of a corrective left shift of one place, with a corresponding change of exponent, in certain circumstances as specified by an auxiliary rule. Perhaps the most obvious way of formulating this auxiliary rule is to have a corrective shift called for when and only when the number of leading zeros would otherwise increase by 1. There are other possibilities, however; one is to make the discrimination on the basis of the normalized factor instead of the product. In this case the corrective shift is ordered whenever the fractional part magnitude

of the normalized factor is less than some specified value, say $\frac{3}{4}$. (Although this rule can give a product with one less or one more leading zero than the smaller factor, its average effect is approximately that desired. Of course, no corrective shift is applied if overflow would result.) The comparative advantages of these two criteria for corrective shift involve some subtlety, and a detailed analysis is needed to choose between them. Both possibilities are considered below, in the discussion of implementation.

If one factor has fractional part 0, then the product of course has fractional part 0, but the exponent should reflect the fact that this value may be only an approximation. It is natural, and turns out to be justifiable, to subtract from the product exponent just that amount necessary to normalize the nonzero factor, with possibly an additional unit decrement for correction. If the other factor has fractional part 0 also, an attempt to normalize it would result in its exponent being decreased by n before the fact became evident. If this is taken to define normalization for zero fractional parts, this case requires no special treatment.

Example (with $n=8$): let two numbers to be multiplied be represented (fractional part and exponent) by

$$f_1 = -.01100010 \qquad e_1 = -100$$
$$f_2 = +.00010010 \qquad e_2 = +001$$

Since $|f_1| > |f_2|$, multiplication is defined in terms of f_1 normalized:

$$f_1' = -.11000100 \qquad e_1' = -101.$$

The 16-digit product $f_1'f_2$, with exponent $e_1'+e_2$, is then:

$$f_1'f_2 = -.00001101\ 11001000 \qquad e_1' + e_2 = -100.$$

If corrective shift is ordered only for $|f_1'| < \frac{3}{4}$, then this is the final form (without round-off). If corrective shift is ordered whenever the product (upper 8 digits) has one more leading zero than the smaller operand f_2, then the shift would be called for in this case. The final product as delivered would be:

$$f_3 = -.00011011 \qquad e_3 = -101.$$

Division

A criterion analogous to that described for multiplication is used in defining the division operation; the first step is to cause the quotient to be adjusted so that it has approximately the same number of significant digits as the operand with the smaller fractional part. If the numerator has the smaller fractional part, the desired effect could be achieved by dividing by the denominator normalized, which is close to 1. If the denominator has the smaller fractional part, it is most simple to conceive of the numerator being first adjusted to the right so that it has the same number of leading zeros as the denominator, and then being divided by the normalized denominator as before. In either case, the quotient has the same number of significant digits as the operand with the fewer such digits, or else one more. Which alternative obtains depends on the relative magnitudes of numerator and denominator, if both are considered normalized. In conventional normalized arithmetic, the extra digit represents overflow, and adjustment must be made. Here it may simply be considered a legitimate part of the quotient unless the operands happen to be in normalized form to begin with. However, as for multiplication, it is advisable to correct the basic result for the tendency to change (here, increase) in number of significant digits. This correction most naturally takes the form of an extra right shift and exponent adjustment whenever the extra digit occurs. The correction then also effects adjustment for actual overflow.

The quotient exponent is basically defined as the difference of the exponents of numerator and denominator. This result must be further adjusted to be consistent with the effective quotient shift. The case of one or both operands 0 is given no special treatment. The result has fractional part 0 and the exponent which is naturally obtained when the above described procedure is carried out.

Example (with $n=8$): let numerator and denominator be represented by:

$$f_1 = -.01100010 \qquad e_1 = -100$$
$$f_2 = +.00010010 \qquad e_2 = +001.$$

Since $|f_1| > |f_2|$, consider f_1 to be adjusted to the right (but without discarding any digits):

$$f_1' = -.00011000\ 10 \qquad e_1' = -010$$

The 8-digit quotient (remainder ignored), with exponent $e_1'-e_2'$ (for normalized denominator, f_2'), is then

$$\frac{f_1'}{f_2'} = -.00101011 \qquad e_1' - e_2' = +000.$$

Since the quotient has one less leading zero than the smaller operand f_2, corrective shift is called for, and the quotient is delivered as:

$$f_3 = -.00010101 \qquad e_3 = +001.$$

Implementation

In this section the realization of the significant digit operations in a parallel binary arithmetic unit is discussed. It is proposed that addition and subtraction be carried out more or less as outlined, by comparing exponents, shifting the fractional part with smaller exponent to the right until the exponents match, then performing the addition or subtraction. If overflow occurs, the customary adjustment is made. Since no further shifting is performed, even when leading zeros appear in the result, the average addition should be less time-consuming than the conventional normalized one. Furthermore, it seems likely that with use of the proposed system there will be a tendency toward less discrepancy among exponents than under the normalization constraint, and this should also decrease the average addition time.

Multiplication and division are both conventionally performed in a series of n steps, counted in the so-called operations counter. At each step a shift is effected, possibly accompanied by an addition (or subtraction) of the multiplicand (or divisor). The pattern of such additions or subtractions is determined by the successive digits of the multiplier (in multiplication) or the successive signs of the partial remainders (in division).

In the multiplication process, it is necessary in some way to obtain the $2n$-digit product shifted to the left by an amount equal to the number of leading zeros in the factor with the larger fractional part; then a correction, consisting of a left shift of one additional place, is applied if circumstances warrant. The basic shift can be accomplished in advance, by normalizing the relevant factor, and then proceeding with a standard n-place multiplication. This has the advantage of procedural convenience, if the correction rule which requires inspection of this factor in normalized form is used. The digits to be examined are then in the leftmost stages of the register in which the factor resides just before the multiplication proper begins. The time taken, however, is the n-step multiplication time plus the shifting time necessary to effect the normalization. The latter time can be saved by doing the multiplication process in a slightly unorthodox manner: let the factor to be normalized be the multiplicand. Then, instead of starting by shifting the partial product *right* at each step, shift the multiplicand *left* until it is normalized. At this point the situation is just as it would have been had the same number of steps been completed with the multiplicand normalized in advance, using the conventional scheme. The multiplication can then proceed in standard fashion from this point on, and the leading digits of the multiplicand can be used to determine whether a correction shift should be performed at the finish (which actually amounts to omitting the last right shift of the multiplication, unless overflow adjustment is required).

Unfortunately, the short-cut described above cannot be employed if the other correction rule is used (the rule which calls for the corrective shift each time the product has one more leading zero than the smaller factor). Unless the multiplicand is normalized in advance, the digit to be inspected may never appear in the leftmost stage of the partial product register, and thus remains virtually inaccessible.

The factor selected as multiplier in the above described processes is that with the greater number of leading zeros, and hence it may be expected that the average number of steps requiring an addition operation (as opposed to a shift only) is minimized. Although customarily this time-saving advantage has been foregone when complement representation for negative numbers is used, because leading zeros become instead leading ones, a multiplication scheme which is symmetric in this respect can be developed, where an addition is ordered whenever the sequence of digits has a change from 1 to 0, a subtraction is ordered when the change is from 0 to 1, and neither is required when there is no change.

Division according to the proposed rules can be accomplished quite naturally in n steps. Here the most convenient way to proceed is to start by shifting left both numerator and denominator, adding 1 to the quotient exponent and counting in the operations counter (equivalent to adding a leading zero to the quotient). When one of the two operands is normalized, the procedure changes; if the denominator is normalized, shifting of the numerator continues, with counting in the operations counter but no change in the quotient exponent. This is continued until the numerator is normalized. Alternatively, if the numerator is normalized first, shifting of the denominator continues, with the exponent increased by 2 at each step, and counting in the operations counter, until the denominator is normalized. When both operands are normalized, the division proper begins; it continues for only that number of steps necessary for the operations counter to reach the full count of n. Note that if the numerator is larger than the denominator after both operands are normalized, the first quotient digit is not an overflow as it would be in normalized arithmetic, unless no preliminary shifting has taken place. However, as mentioned previously, an extra shift right should be performed in this case anyway as a correction. A scheme for speeding up the process, by ordering an arithmetic operation (subtraction or addition) only when the sequence of digits in the partial remainder changes from 0 to 1 or from 1 to 0, is available for division as it is for multiplication.

The case of zero operands is included automatically in these implementation schemes if the attempt to normalize is always halted after n steps.

Conclusion

Significant digit arithmetic seems to offer a means of analyzing floating point calculations more effectively, and also to possess certain advantages from the equipment standpoint. It is planned to use this system on the Maniac III computer, now under construction at the Institute for Computer Research of the University of Chicago.

Minimal "Sum of Products of Sums" Expressions of Boolean Functions*

SHREERAM ABHYANKAR†

Summary—The problem of economical synthesis of circuits for digital computers leads to the problem of finding Boolean expressions of minimal length equivalent to a Boolean expression f. Previous authors restricted themselves to "sum of products" expressions; dualizing this gives "products of sums." The next more efficient step is to find minimal "sums of products of sums" expressions. In this paper, the basic concepts are formulated in Part I, and general theorems are given in Part II. In Part III, all the distinct minimal "sum of products of sums" expressions are obtained in case the cell complex of f consists of two isolated points. For the case of three isolated points partial results have been obtained which will be published in a later communication.

I. Introduction

Section 1

WE START by fixing our notations. By a Boolean expression $f(x_1, x_2, \cdots, x_n)$ in the letters $x_1, x_2, \cdots, x_n$ will be meant an expression in $x_1, x_2, \cdots, x_n$ and their complements $x_1', x_2', \cdots, x_n'$ joined by a certain number of product and sum operations. We can render this more precise as follows: a literal, *i.e.*, an expression either of type x_i or of type x_i', will be called a Boolean expression of *zero iteration;* we shall follow the usual custom of denoting x_i and x_i' by x_i^1 and x_i^0, respectively. Expressions of type

$$\prod_{k=1}^{m} x_{i_k}^{j_k} = x_{i_1}^{j_1} x_{i_2}^{j_2} \cdots x_{i_m}^{j_m}, \qquad m \geqq 1,$$

and of type

$$\sum_{k=1}^{m} x_{i_k}^{j_k} = x_{i_1}^{j_1} + x_{i_2}^{j_2} + \cdots + x_{i_m}^{j_m}, \qquad m \geqq 1,$$

will be called Boolean expressions of *one iteration*, respectively, of type *product* (in short p) and of type *sum* (in short s); here i_k and j_k take values in the sets $(1, 2, \cdots, n)$ and $(0, 1)$, respectively. By induction we define a Boolean expression of $(q+1)$ iterations of type *product-sum-product-* $\cdots$ (in short $psp \cdots$) to be an expression of type

$$\prod_{k=1}^{m} g_k(x_1, \cdots, x_n) = g_1(x_1, \cdots, x_n) g_2(x_1, \cdots, x_n) \cdots g_m(x_1, \cdots, x_n),$$

and a Boolean expression of $(q+1)$ iterations of type *sum-product-sum-* $\cdots$ (in short $sps \cdots$) to be an expression of type

$$\sum_{k=1}^{m} g_k(x_1, x_2, \cdots, x_n) = g_1(x_1, \cdots, x_n) + \cdots + g_m(x_1, \cdots, x_n),$$

where $g_1(x_1, \cdots, x_n), \cdots, g_m(x_1, \cdots, x_n)$ with $m \geqq 1$ are Boolean expressions of q-iterations of types $sps \cdots$ and $psp \cdots$, respectively. A Boolean expression $f(x_1, \cdots, x_n)$ is then a Boolean expression of some iterations. Let $f(x_1, \cdots, x_n)$ be a Boolean expression of q-iterations. Observe, that if q is even then f is either of type

$$sp\ sp \cdots sp(\text{"}sp\text{" repeated } q/2 \text{ times}) \cdots, \qquad (1)$$

or of type

$$ps\ ps \cdots ps\ (\text{"}ps\text{" repeated } q/2 \text{ times}) \cdots; \qquad (2)$$

and if q is odd then f is either of type

$$sp\ sp \cdots sp\ s(\text{"}sp\text{" repeated } (q-1)/2 \text{ times}), \qquad (3)$$

or of type

$$ps\ ps \cdots ps\ p(\text{"}ps\text{" repeated } (q-1)/2 \text{ times}). \qquad (4)$$

If h is any positive even integer then f can be considered to be a Boolean expression of $q+h$ iterations, respectively, of type

$$sp\ sp \cdots sp\ (\text{"}sp\text{" repeated } (q+h)/2 \text{ times}), \qquad (1^*)$$

$$ps\ ps \cdots ps\ (\text{"}ps\text{" repeated } (q/h)/2 \text{ times}), \qquad (2^*)$$

$$sp\ sp \cdots sp\ s(\text{"}sp\text{" repeated } (q+h-1)/2 \text{ times}), \qquad (3^*)$$

or

$$ps\ ps \cdots ps\ p\ (\text{"}ps\text{" repeated } (q+h-1)/2 \text{ times}), \qquad (4^*)$$

according as f is of type (1), (2), (3), or (4); similarly, if h is any positive odd integer then f can be considered to be of $q+h$ iterations of a corresponding type.

Note that in Boolean expressions, addition and multiplication are to be commutative, thus, for instance,

$$x_1x_2 + x_3x_4' = x_4'x_3 + x_1x_2.$$

By a *general* Boolean expression of 0 or 1 iteration will be meant a Boolean expression of 0 or 1 iteration, respectively; by induction we define a *general* Boolean expression of $q+1$ iterations to be an expression of type

$$f_1(x_1, \cdots, x_n) + \cdots + f_m(x_1, \cdots, x_n),$$

* Manuscript received by the PGEC, December 17, 1957; revised manuscript received, May 20, 1958. First part of a report entitled "Investigation of Mathematical Methods for the Analysis and Synthesis of Computer Circuits," submitted to AF Cambridge Res. Center, under Contract No. AF 19(604)-1818, Columbia University, New York, N. Y., September, 1956.

† Princeton University, Princeton, N. J.

or of type

$$f_1(x_1, \cdots, x_n)f_2(x_1, \cdots, x_n) \cdots f_m(x_1, \cdots, x_n),$$

or of type

$$[f(x_1, x_2, \cdots, x_n)]',$$

where $f_1, f_2, \cdots, f_m, f$ are general Boolean expressions of i iterations. By the length of a general Boolean expression $f(x_1, \cdots, x_n)$, to be denoted by $L(f(x_1, \cdots, x_n))$, will be meant the number of sum and product operations in $f(x_1, \cdots, x_n)$.

Let $f(x_1, \cdots, x_n)$ be a general Boolean expression. It naturally gives rise to a Boolean function which we shall denote by $\bar{f}(x_1, \cdots, x_n)$ of the *binary variables* $x_1, x_2, \cdots, x_n$ in the usual sense; when there is no confusion, we shall replace $\bar{f}(i_1, \cdots, i_n)$ by $f(i_1, \cdots, i_n)$, where $(i_1, \cdots, i_n)$ is a vector with components 0 or 1. Let $g(x_1, \cdots, x_n)$ be another general Boolean expression. We shall say that the Boolean expressions $f(x_1, \cdots, x_n)$ and $g(x_1, \cdots, x_n)$ are *equivalent*, in symbols $f(x_1, \cdots, x_n) \sim g(x_1, \cdots, x_n)$, if the corresponding Boolean functions $\bar{f}(x_1, \cdots, x_n)$ and $\bar{g}(x_1, \cdots, x_n)$ are equal.

Informally speaking, $f(x_1, \cdots, x_n) \sim g(x_1, \cdots, x_n)$ if, and only if, f can be converted into g by a finite number of applications of the various Boolean laws.

Given a general Boolean expression $f(x_1, \cdots, x_n)$, by applications of the laws of complementation we can at once find a Boolean expression $F(x_1, x_2, \cdots, x_n)$, such that $f(x_1, \cdots, x_n) \sim F(x_1, x_2, \cdots, x_n)$, and we shall have $L(f) = L(F)$. Hence, without any loss of generality we may, and henceforth we shall, confine our attention to what we have called *Boolean expressions* (and not to the more general *general Boolean expressions*).

Given a Boolean expression $f(x_1, x_2, \cdots, x_n)$, the Boolean expression obtained from $f(x_1, x_2, \cdots, x_n)$ by interchanging x_i and x_i' for $i = 1, \cdots, n$ and interchanging *sum* and *product* will be called the Boolean expression complementary to $f(x_1, \cdots, x_n)$ and will be denoted by $\tau f(x_1, \cdots, x_n)$.

Now we shall illustrate the various concepts introduced in this section by taking examples in five variables.

(*A*):

1) x_1, x_3' are Boolean expressions of zero iteration.
2) x_1+x_4, $x_3+x_1+x_5$ are Boolean expressions of 1 iteration and of type *s*.
3) $x_3x_4(x_1+x_2')$ is a Boolean expression of 2 iterations and of type *ps*. It can also be considered as a Boolean expression of 3 iterations and type *sps* or of type *psp*; also as a Boolean expression of 4 iterations and of type *psps*; etc.
4) $(x_1x_2+x_2'x_5)(x_1x_4x_5+x_3')(x_2+x_3+x_4)(x_3'x_5x_1')$ is a Boolean expression of 3 iterations and of type *psp*.
5) $[(x_1+x_3+x_4)(x_2+x_5')(x_4+x_1)+(x_3+x_4)x_5x_1']$ $[x_3+x_1]$ is a Boolean expression of 4 iterations and type *psps*.

(*B*):

$(x_3+x_1)'+x_4x_5+(x_1x_2)'$ is a general Boolean expression of length 5 and it is equivalent to the Boolean expression $x_3'x_1'+x_4x_5+x_1'+x_2'$ of the same length which is obtained from the original one by the use of the law of complementation, *i.e.*, setting: $(x_3+x_1)' = x_3'x_1'$ and $(x_1x_2)' = x_1'+x_2'$.

(*C*):

Let

$$f(x_1, x_2, x_3, x_4, x_5) = x_1(x_2 + x_3')$$

and

$$g(x_1, x_2, x_3, x_4, x_5) = x_1x_2 + x_1x_3'.$$

Then

$$\bar{f}(1, 0, 0, 0, 0) = 1(0 + 0') = 1(0 + 1) = 1 \cdot 1 = 1$$

and

$$\bar{g}(1, 0, 0, 0, 0) = 1 \cdot 0 + 1 \cdot 0' = 0 + 1 \cdot 1 = 0 + 1 = 1.$$

Similarly for all the $2^5 = 32$ vectors it can be verified that the value of $\bar{f}$ equals the value of $\bar{g}$. Hence $\bar{f} = \bar{g}$ so that $f \sim g$. This can be seen by applying the distributive law to f and thus obtaining g. Note that $L(f) = 2$ and $L(g) = 3$.

(*D*):

Let

$$f(x_1, x_2, x_3, x_4, x_5) = x_1x_2 + x_3(x_1 + x_2') + x_5'(x_3 + x_2').$$

Then

$$\tau f(x_1, x_2, x_3, x_4, x_5) = (x_1' + x_2')(x_3' + x_1'x_2)(x_5 + x_3'x_2).$$

Section 2

For a given Boolean expression $f(x_1, x_2, \cdots, x_n)$ we shall denote by $Y(f)$ the set of all the Boolean expressions in $x_1, x_2, \cdots, x_n$ which are equivalent to f; by $Ysp(f)$, $Yps(f)$ and $Ysps(f)$ we shall denote the sets of all the Boolean expressions in $x_1, x_2, \cdots, x_n$ which are equivalent to f, and which are of type sum-product, product-sum and sum-product-sum, respectively; by $LZ(f)$, $LZsp(f)$, $LZps(f)$ and $LZsps(f)$ we shall denote the minimum of $L(g)$ for g in $Y(f)$, $Ysp(f)$, $Yps(f)$, and $Ysps(f)$, respectively; finally by $Z(f)$, $Zsp(f)$, $Zps(f)$ and $Zsps(f)$ we shall denote the set of members of $Y(f)$, $Ysp(f)$, $Yps(f)$ and $Ysps(f)$, which are of length $LZ(f)$, $LZsp(f)$, $LZps(f)$ and $LZsps(f)$, respectively. Observe that

$$Y(f) \supset Ysps(f), \quad Ysps(f) \supset Ysp(f), \quad Ysps(f) \supset Yps(f),$$

and

$$LZ(f) \leqq LZsps(f) \leqq \min [LZsp(f), LZps(f)].$$

Also note that since there are only a finite number of distinct Boolean expressions of a given length, $Z(f)$, $Zsps(f)$, $Zsp(f)$ and $Zsp(f)$ are finite sets.

The importance of the problem of finding Boolean expressions $g(x_1, \cdots, x_n)$ equivalent to a given Boolean expression $f(x_1, x_2, \cdots, x_n)$ such that $L(g)$ is minimal; *i.e.*, the problem of finding $Z(f)$ and calculating $LZ(f)$ is due to its interpretations as *the problem of simplifying truth functions* and as *the problem of economical synthesis of circuits for digital computers*. But concerning this absolute minimization problem, due to its very difficult nature, up until now nothing has been done. Various authors have exclusively restricted their attention to the much simpler problem of finding minimal normal expressions, *i.e.*, in our terminology that of finding $Zsp(f)$; among these authors prominent have been Quine,[1,2] Urbano and Mueller,[3] and Petrick.[4] The methods developed by them for finding $Zsp(f)$ are quite good, and we at once observe that by applying their methods to the complement τf of the given expression f, one can find $Zps(f)$. $Zps(f)$ coincides with the set of complements of members of $Zsp(\tau f)$. As we shall presently illustrate, even with expressions in as few as four letters, we may have $LZ(f) < \min [LZsp(f), Lps(f)]$. We take the example considered by Mueller and Urbano[3] on page 129 but we label our letters as x_1, x_2, x_3, x_4 instead of x_0, x_1, x_2, x_3.

$$f(x_1, x_2, x_3, x_4) = x_1'x_2'x_3'x_4' + x_1x_2'x_3'x_4' + x_1x_2x_3'x_4' + x_1'x_2'x_3x_4' + x_1x_2'x_3x_4' + x_1'x_2x_3x_4' + x_1x_2x_3'x_4 + x_1'x_2x_3x_4 + x_1x_2x_3x_4.$$

The elements of $Zsp(f)$ given on page 130 of Urbano and Mueller have length 10, *i.e.*, we have

$$LZsp(f) = 10.$$

To find $Zps(f)$ we proceed by observing:

$$\tau f(x_1, x_2, x_3, x_4) \sim f'(x_1, x_2, x_3, x_4) = x_1'x_2'x_3x_4 + x_1'x_2'x_3'x_4 + x_1'x_2x_3'x_4' + x_1'x_2x_3'x_4 + x_1x_2'x_3x_4 + x_1x_2'x_3'x_4 + x_1x_2x_3x_4'.$$

Let us plot f' on a four cube. (See Fig. 1.) Now each of the basic cells I, II, III covers a vertex not covered by any other basic cell, and together they cover the entire complex.[5] Hence, there is only one irredundant normal form: $g(x_1, x_2, x_3, x_4) = x_1x_2x_3x_4' + x_1'x_2x_3' + x_2'x_4$ which is then the only member of $Zsp(f')$. Hence,

$$\tau g(x_1, x_2, x_3, x_4) = (x_1' + x_2' + x_3' + x_4)(x_1 + x_2' + x_3)(x_2 + x_4')$$

is the only member of $Zps(f)$ and $LZps(f) = L(\tau g) = 8$. Thus,

$$LZsp(f) = LZps(\tau f) = 10 \text{ and } LZps(f) = LZsp(\tau f) = 8.$$

Now let

$$k(x_1, x_2, x_3, x_4) = [x_2' + (x_1' + x_3' + x_4)(x_1 + x_3)](x_2 + x_4').$$

Then $k \sim \tau g \sim f$ and $L(k) = 7$. Therefore,

$$LZ(f) \leqq LZpsps(f) \leqq 7 < \min [LZsp(f), LZps(f)] = 8,$$

and

$$LZsps(f) \leqq \min [LZsp(f), LZps(f)] = 8 < LZsp(f) = 10.$$

It is clear that when we are dealing with Boolean expressions in a large number of letters, $LZ(f)$ will in general be much smaller than $Zsp(f)$ as well as $Zps(f)$.

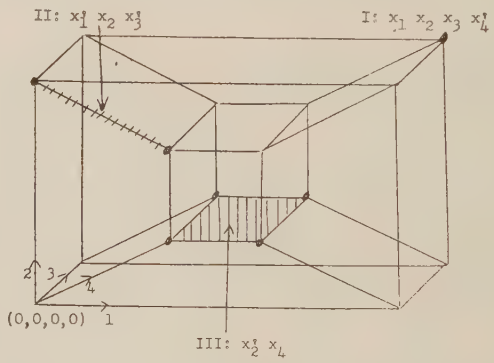


Fig. 1

Section 3

Thus, the minimizing efficiency yielded by $Zsps(f)$ is at least as much as the minimizing efficiencies of $Zsp(f)$ and $Zps(f)$ put together. Of course the most efficient minimizing will consist in developing methods for finding the absolute minimal expressions, *i.e.* $Z(f)$. As we have already stated, nothing has yet been done in either of these two directions; hence any sort of beginning, howsoever modest, is a worthwhile goal. To contribute to the study of $Zsps(f)$ is the purpose of the present paper, while a contribution to the much more difficult study of $Z(f)$ will be made in a paper which will follow this one.[6]

Section 4

As in Urbano and Mueller[3] with a Boolean expression in n letters we may associate its cell complex in the n-cube.[5] It is reasonable to start by studying those cell

[1] W. V. Quine, "The problems of simplifying truth functions," *Amer. Math. Monthly*, vol. 59, pp. 521–531; October, 1952.

[2] W. V. Quine, "A way to simplify truth functions," *Amer. Math. Monthly*, vol. 62, pp. 627–631; November, 1955.

[3] R. H. Urbano and R. K. Mueller, "A topological method for the determination of the minimal forms of a Boolean function," IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-5, pp. 126–132; September, 1956.

[4] S. R. Petrick, "A direct determination of the irredundant forms of a Boolean function from the set of prime implicants," Computer Lab. A.F.C.R.C. Bedford, Mass.; April, 1956.

[5] We shall use, without explicit reference, the geometric notions and terminology introduced by Urbano and Mueller, footnote 3.

[6] S. Abhyankar, "Absolute minimal expressions of Boolean functions," in preparation.

complexes which possess nice geometrical properties, *e.g.* purity of dimension (*i.e.* all the basic cells[5] being of a same dimension), connectedness, etc.; first because natural problems are to be expected to yield configurations with nice properties, and second, by decomposing a given cell complex into nice components, one can use the theory of special configurations towards the minimization of the general case. We propose to start with pure zero dimensional complexes, *i.e.*, complexes which consist of a certain number of isolated points. Another justification for studying low dimensional (dimension of a complex = the maximum dimension of a basic cell occurring in it)—in particular zero dimensional–complexes is that in a random given set of vertices the probability of the existence of an s-dimensional cell drops rapidly as s increases.

In this paper we shall study complexes which consist of one or two isolated points in an arbitrary dimensional cube (*i.e.*, no restriction on the number of letters of the Boolean expressions under consideration). We remark that the importance of our present investigations perhaps lies more in the methods used in the proofs than in the results themselves. Throughout the paper, we illustrate the results by examples in a small number of variables.

II. Generalities

Section 5

Let $f(x_1, \cdots, x_n)$ be a Boolean expression. We shall denote the cell complex (on the n-cube S_n) of $f(x_1, \cdots, x_n)$ (as considered by Urbano and Mueller[5]) by $C(f)$ and the point set consisting of the points of $C(f)$ by $D(f)$, and we shall call $D(f)$ the *point of complex* of f. Recall that the points of $D(f)$ are in one to one correspondence with the terms of the canonical Boolean expression

$$F(x_1, \cdots, x_n) = \sum f(i_1, i_2, \cdots, i_n) x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n}$$

equivalent to $f(x_1, \cdots, x_n)$, (the sum is taken over all the 2^n vectors $(i_1, i_2, \cdots, i_n)$ with coordinates 0 or 1). Consequently, a statement about the point-complex $D(f)$ is simply the geometric equivalent of an arithmetic statement about F, and hence we can and will use these two modes of studying $f(x_1, \cdots, x_n)$ interchangeably. For a point set E of S_n, we shall denote by $\tau(S_n)E$ the complement of E in S_n, when the reference to S_n is clear from the context $\tau(S_n)E$ will be replaced by τE.

Let

$$g(x_1, \cdots, x_n) = \sum_{k=1}^{m} g_k(x_1, \cdots, x_n)$$

be a Boolean expression where $g_1, \cdots, g_m$ are Boolean expressions of type $ps \cdots$. Then

$$D(g) = \bigcup_{k=1}^{m} D(g_k) \quad \text{and} \quad g \sim f$$

if, and only if, $D(g) = D(f)$.

Now assume that $g_1, \cdots, g_m$ are of type product-sum so that g is of type sum-product-sum; let $E_k = D(g_k)$. Then we have that $g \in Ysps(f)$ if, and only if,

$$\bigcup_{k=1}^{m} E_k = D(f); \quad \text{and} \quad g \in Zsps(f)$$

if, and only if,

$$\bigcup_{k=1}^{m} E_k = D(f) \quad \text{and} \quad L(g) = m - 1 + \sum_{k=1}^{m} L(g_k)$$

is a minimum, *i.e.*, if, and only if, for $k = 1, \cdots, m$, g_k is in Zps (E_k),

$$\bigcup_{k=1}^{m} E_k = D(f) \quad \text{and} \quad m - 1 + \sum_{k=1}^{m} LZps(E_k)$$

is a minimum; furthermore when the latter is a minimum and g_k' is any member of $Zps(E_k)$, then

$$L\left(\sum_{k=1}^{m} g_k'\right) = m - 1 + \sum_{k=1}^{m} L(g_k')$$

$$= m - 1 + \sum_{k=1}^{m} LZps(E_k) = L(g) = LZsps(f),$$

and hence $\sum_{k=1}^{m} g_k'$ is also in $Zsps(f)$. Now $LZps(E_k) = LZsp(\tau E_k)$. Therefore, the problem of finding $Zsps(f)$ consists in finding partitions $E_1, \cdots, E_m$ of $D(f)$, for which

$$m - 1 + \sum_{k=1}^{m} LZsp(\tau E_k)$$

is minimum, and finding $Zsp(\tau E_1), \cdots, Zsp(\tau E_m)$; let $E_1, \cdots, E_m$ be any such partition of $D(f)$ and let h_k be any member of $Zsp(\tau E_k)$, then $\sum_{k=1}^{m} \tau h_k$ is in $Zsps(f)$, and all the members of $Zsps(f)$ are obtained in this manner.

Section 6

We introduce the following notation. Consider a Boolean expression $f(x_1, \cdots, x_n)$ in letters $x_1, \cdots, x_n$ which is of the product type, *i.e.*, $f(x_1, \cdots, x_n)$ is the product of a certain number of x_i and a certain number of x_j'. Let S_n be the n-cube of the letters $x_1, \cdots, x_n$. Then $D(f)$ consists of the points of a certain subcube S_m (*i.e.*, a cell) of S_n (the subscripts of S_m and S_n denote their dimensions). By $I(f)$ or $I(S_m)$ we shall denote the set of subscripts i for which x_i occurs in f and x_i' does not; by $J(f)$ or $J(S_m)$ we shall denote the set of subscripts j for which x_j' occurs in f and x_j does not. Finally by $K(1, 2, \cdots, n)$ (f) or $K(1, \cdots, n)(S_m)$ we shall denote the complement of $I(f) \cup J(f)$ in $(1, 2, \cdots, n)$; when the reference to $(1, 2, \cdots, n)$ is clear from the context we shall use the shorter notations $K(f)$ or $K(S_m)$.

Section 7

Finally we are ready to prove a theorem. Let $f(x_1, \cdots, x_n)$ be a Boolean expression in letters $x_1, \cdots, x_n$ and consider the point-complex $D(f)$ of f in the n-cube S_n. Let S_m be an m-cell of S_n which contains $D(f)$; in particular S_m could be the subcube of S_n spanned by the points of $D(f)$, *i.e.*, the smallest cell of S_n containing $D(f)$. Let

$$(k_1, k_2, \cdots, k_m) = K(1, 2, \cdots, n)(S_m).$$

Then as a subset of S_m we can think of $D(f)$ as giving a canonical Boolean expression $F(x_{k_1}, x_{k_2}, \cdots, x_{k_m})$ in the m letters $x_{k_1}, \cdots, x_{k_m}$. Let

$$H(x_1, \cdots, x_n) = \prod_{i \in I(S_m)} Xi \prod_{j \in J(S_m)} x_j',$$

so that $H(x_1, \cdots, x_n)$ defines S_m in S_n.

Theorem 1: Let $G(x_{k_1}, \cdots, x_{k_m})$ be a Boolean expression in $Zps(F(x_{k_1}, \cdots, x_{k_m}))$. Then

$$g(x_1, \cdots, x_n) = G(x_{k_1}, \cdots, x_{k_m})H(x_1, \cdots, x_n)$$

is in $Zps(f(x_1, \cdots, x_n))$. Furthermore $G \leftrightarrow g$ is a one to one correspondence between the members of $Zps(F)$ and members of $Zps(f)$.

Proof: We may relabel the letters so that

$$K(S_m) = (1, 2, \cdots, m),$$
$$I(S_m) = (m+1, m+2, \cdots, m+s)$$

and

$$J(S_m) = (m+s+1, m+s+1, \cdots, m+s+t = n).$$

Let $A = \tau(S_n)(S_m)$ and $B = \tau(S_m)(D(f)) = \tau(S_m)(D(F))$. Then A and B are disjoint and

$$D(\tau f) = \tau(S_n)(D(f)) = A \cup B.$$

Let X_i and X_j' be the $(n-1)$ cells of S_n with equations $x_i = 1$ and $x_j = 0$, respectively. Then $X_{m+1}', X_{m+2}', \cdots, X_{m+s}', X_{m+s+1}, X_{m+s+2}, \cdots, X_n$ are contained in A, and hence they are all basic cells of $C(\tau f)$. We are assuming that f is not inconsistent, *i.e.*, $D(f)$ is not empty so that $m > 0$, and hence S_n is not in $C(\tau f)$. Now the point of S_n given by

$$x_1 = x_2 = \cdots = x_m = 1, x_{m+1} = 0, x_{m+2} = x_{m+3} = \cdots$$
$$= x_{m+s} = 1, x_{m+s+1} = x_{m+s+2} = \cdots = x_n = 0$$

(*i.e.*, the point which corresponds to $x_1 \cdots x_m x_{m+1}'$, $x_{m+2}x_{m+3} \cdots x_{m+s}x_{m+s+1}' \cdots x_n'$) is in X_{m+1}' but not in any of the remaining above mentioned basic cells of $C(\tau f)$, and hence X_{m+1}' forms an essential star[5] of $C(\tau f)$. Similarly, each of the above mentioned basic cells of $C(\tau f)$ is an essential star of $C(\tau f)$. Hence, in view of the results of Urbano and Mueller,[3] and Petrick,[4] the minimal coverings of $D(\tau f)$ consist of the minimal coverings of B adjoined by the cells $X_{m+1}', X_{m+2}', \cdots, X_{m+s}', X_{m+s+1}, \cdots, X_n$. Arithmetizing this geometrical statement we conclude the following. If $G(x_1, \cdots, x_m)$ is a member of $Zsp(\tau F)$, then

$$g(x_1, \cdots, x_n) = G(x_1, \cdots, x_m) + x_{m+1}' + x_{m+2}' + \cdots + x_{m+s}' + x_{m+s+1} + x_{m+s+2} + \cdots + x_n$$

is a member of $Zsp(\tau f)$, and $G \leftrightarrow g$ is a one to one correspondence between the members of $Zsp(\tau F)$ and the members of $Zsp(\tau f)$. Complementing this result we at once obtain the required one to one correspondence between $Zps(F)$ and $Zps(f)$.

Example: Take $n = 6$ and let

$$f(x_1, c_2, x_3, x_4, x_5, x_6) = x_5'x_1x_3'x_6 + x_2x_3'x_6 + x_3'(x_4x_6 + x_3x_5).$$

Then applying the distributive and cancellation laws we have

$$f(x_1, x_2, x_3, x_4, x_5, x_6) \sim x_1x_3'x_5'x_6 + x_2x_3'x_6 + x_3'x_4x_6 + x_3'x_3x_5 \sim x_1x_3'x_5'x_6 + x_2x_3'x_6 + x_3'x_4x_6.$$

In each term of the last expression, x_3' and x_6 are common. This means that the point-complex $D(f)$ of f in the 6-cube S_6 is contained in the 4-cube S_4 represented by the expression $H(x_1, x_2, x_3, x_4, x_5, x_6) = x_3'x_6$. We have: $I(S_4) = (6)$, $J(S_4) = (3)$, $K(1, 2, 3, 4, 5, 6)(S_4) = (1, 2, 4, 5)$ Factoring out $x_3'x_6$ from f we set $F(x_1, x_2, x_4, x_5)$ $x_1x_5' + x_2 + x_6$. Then the point-complex of F considered as a subset of S_4 coincides with the point-complex of f considered as a subset of S_6. Theorem 1 now says that if $G(x_1, x_2, x_4, x_5)$ is a minimal *ps* expression equivalent to $F(x_1, x_2, x_4, x_5)$, then $g(x_1, x_2, x_3, x_4, x_5, x_6) = G(x_1, x_2, x_4, x_5)x_3'x_6$ is a minimal *ps* expression equivalent to $f(x_1, x_2, x_3, x_4, x_5, x_6)$, and each such expression is obtained in this manner.

Theorem 2: For the Boolean expression $f(x_1, \cdots, x_n) = x_{i_1} \cdots x_{i_s} x_{j_1}' \cdots x_{j_t}'$, where the subscripts $i_1, \cdots, i_s, j_1, \cdots, j_t$ are all distinct $[m = n - s - t \geqq 0$, so that f represents an m dimensional cell of S_n; in particular if $m = 0$ then it is a point$]$, we have that f is the only member of $Zps(f)$ and $LZps(f) = n - m - 1$.

Proof: In Theorem 1 take S_m to be the cell defined by f.

Theorem 3: For the Boolean expression $f(x_1, \cdots, x_n) = x_{i_1} \cdots x_{i_s}, \; x_{j_1}' \cdots x_{j_t}'$, where the subscripts $i_1, \cdots, i_s, j_1, \cdots, j_t$ are all distinct, and $s + t = n$, so that $D(f)$ is a single point; we have that f is the only member of $Zsps(f)$, so that $Zsps(f) = Zps(f)$ and $LZsps(f) = LZps(f) = n - 1$.

Proof: Let $g = g_1 + g_2 + \cdots + g_q$ be a member of $Zsps(f)$ where $g_1, g_2, \cdots, g_q$ are of type *ps*. Then

$$\bigcup_{k=1}^{q} D(g_k) = D(f).$$

Since $D(f)$ is a single point we must have $D(g_k) = D(f)$ for some k, say $D(g_1) = D(f)$. The minimality of g implies that $g = g_1$ so that $q = 1$ and g is of type *ps*. Now Theorem 3 follows from Theorem 2.

III. Case of Two Isolated Points

Section 8

Consider the Boolean expression

$$f(x_1, \cdots, x_n) = (x_1x_2 \cdots x_n + x_1'x_2' \cdots x_n')', \quad n > 1.$$

In this section we propose to find the minimal sum-product equivalents of f, *i.e.*, the members of $Zsp(f)$. As usual, let $C(f)$ be the cell complex of f in the n-cube S_n

For $i \neq j$ let A_{ij} denote the $(n-2)$-cell of S_n defined by x_ix_j', *i.e.*, by: $x_i=1$ and $x_j=0$. Now $x_1x_2 \cdots x_n$ is the product of all the n unprimed letters and $x_1'x_2' \cdots x_n'$ is the product of all the n primed letters, and hence neither is contained in the $A_{ij}=x_ix_j'$ which has one unprimed letter and one primed letter. Therefore, A_{ij} is a cell of $C(f)$. Now the only $(n-1)$-cells which contain A_{ij} are x_i and x_j'; x_i contains the point $x_1x_2 \cdots x_n$ of $\tau D(f)$ and x_j' contains the point $x_1'x_2' \cdots x_n'$ of $\tau D(f)$ so that neither x_i nor x_j' is in $C(f)$. Hence, A_{ij} is a basic cell of $C(f)$.

Now let E be any cell of $C(f)$. If $I(E)$ were empty then the point $x_1'x_2' \cdots x_n'$ of $\tau D(f)$ would be in E, and if $J(E)$ were empty then the point $x_1x_2 \cdots x_n$ of $\tau D(f)$ would be in E, each of these implications is a contradiction, and hence neither $I(E)$ nor $J(E)$ is empty. Fix i in $I(E)$ and j in $J(E)$, then $E \subset A_{ij}$. Thus we have proved

Lemma 1: The $n(n-1)$ cells A_{ij} with $i \neq j$ (each of which is $n-2$ dimensional) are exactly all the basic cells of $C(f)$.

One consequence of Lemma 1 is that any irreducible covering of $D(f)$ consists of a certain number of the cells A_{ij}. Let H be a set of ordered pairs (ij) such that the cells A_{ij} with (ij) in H form a covering of $D(f)$.

Lemma 2: There exists j with (nj) in H.

Proof: Let H_1 be the set of members of H of type (in), and let H_2 be the set of the remaining members of H of type (ij) with $j \neq n$.

Since $f=(x_1x_2 \cdots x_n+x_1'x_2' \cdots x_n')'$, in the canonical expression equivalent to f occur all the possible products $x_1^{k_1}x_2^{k_2} \cdots x_n^{k_n}$ such that not all the k_t are zero and not all the k_t are one, *i.e.*, at least one of the k_t is 0 and at least one of the k_t is 1; let us denote by T the set of all these products [T is essentially the same as $D(f)$].

For each (ij) in H we write

$$x_ix_j' \sim x_ix_j'(\sum x_1^{k_1}x_2^{k_2} \cdots \hat{x}_i \cdots \hat{x}_j \cdots x_n^{k_n})$$
$$\sim \sum x_1^{k_1}x_2^{k_2} \cdots x_i^{1} \cdots x_j^{0} \cdots x_n^{k_n} \qquad (1_{ij})^7,$$

where $\hat{}$ denotes omission and where the summations are extended over all the 2^{n-2} combinations of $k_t=0$ or 1 with $t=1, 2, \cdots, \hat{i}, \cdots, \hat{j}, \cdots, n$. Since

$$f \sim \sum_{(ij) \in H} x_ix_j',$$

if we collect all the terms in summations (1_{ij}) for the various (ij) in H, we must end up with exactly the terms in T.

Assume, if possible, that there does not exist any j with (nj) in H. Then any term in H_2 is of type (ij) with $i \neq n$ and $j \neq n$. For a term of this type, the summation (1_{ij}) contains $x_1^{k_1} \cdots x_{n-1}^{k_{n-1}} x_n$ if, and only if, it contains $x_1^{k_1} \cdots x_{n-1}^{k_{n-1}} x_n'$. Since $x_1'x_2' \cdots x_n'$ does not belong to T, it cannot be in (1_{ij}), and hence $x_1'x_2' \cdots x_{n-1}' x_n$ cannot belong to (1_{ij}).

Thus $x_1'x_2' \cdots x_{n-1}' x_n$ does not come from any term in H_2, but it does belong to T, and hence it must come from some term in H_1, *i.e.*, a term of type (in). But each product in (1_{in}) contains x_n as a primed letter, and hence $x_1'x_2' \cdots x_{n-1}' x_n$ cannot come from any term in H_1. This is a contradiction which proves our assertion.

Lemma 3: For any i with $1 \leqq i \leqq n$, there exists j with (ij) in H.

Proof: Due to the symmetry in the subscripts $1, 2, \cdots, n$ it is enough to treat the case of any one particular value of i, say $i=n$. But for $i=n$ Lemma 3 reduces to Lemma 2.

Lemma 4: For any j with $1 \leqq j \leqq n$, there exists i with (ij) in H.

Proof: In view of the symmetry between the primed and the unprimed letters, Lemma 4 follows from Lemma 3.

Thus Lemmas 3 and 4 imply that each of the $2n$ literals $x_1, \cdots, x_n, x_1', \cdots, x_n'$ occurs at least once in the expression $\sum_{(ij) \in H} x_ix_j'$. Let h be the number of terms in H. Then $2h \geqq 2n$, *i.e.*,

Lemma 5:

$$h \geqq n.$$

Lemma 6: Let

$$g(x_1, \cdots, x_n) = x_1x_2' + x_2x_3' + \cdots + x_{n-1}x_n' + x_nx_1'.$$

Then $g \sim f$.

Proof: Since g is the sum of a certain number of cells A_{ij}, we have $D(g) \subset D(f)$, and hence it is enough to prove that $D(f) \subset D(g)$. Let $x_1^{k_1}x_2^{k_2} \cdots x_n^{k_n}$ be a point of $D(f)$. Then k_t is zero for some t and it is 1 for some t. Let u be the minimum value of t such that $k_t=1$, and let v be the maximum value of t such that $t \geqq u$ and $k_u=k_{u+1}= \cdots =k_t=1$. If $v=n$ than $k_1=0$ so that $x_1^{k_1}x_2^{k_2} \cdots x_n^{k_n}$ is in x_nx_1', and if $v<n$ then $k_{v+1}=0$ so that $x_1^{k_1}x_2^{k_2} \cdots x_n^{k_n}$ is in x_vx_{v+1}'. Therefore, $x_1^{k_1}x_2^{k_2} \cdots x_n^{k_n}$ is in $D(g)$.

Lemma 7: $x_1x_2'+x_2x_3'+ \cdots +x_{n-1}x_n'+x_nx_1'$ is in $Zsp(f)$ and $LZsp(f)=2n-1$.

Proof: Follows from Lemmas 5 and 6.

Now assume H is such that $\sum_{(ij) \in H} x_ix_j'$ is in $Zsp(f)$. Then Lemmas 5 and 7 tell us that H contains n pairs. Lemmas 3 and 4 tell us that given i there exists j with (ij) in H. Hence if we label elements of H as (i_kj_k) with $k=1, 2, \cdots, n$ then $(i_1, i_2, \cdots, i_n)$ and $(j_1, j_2, \cdots, j_n)$ are both rearrangements of $(1, 2, \cdots, n)$. Obviously

[7] This is simply a numbering of these summations.

H is completely characterized by the permutation

$$H^* = \begin{pmatrix} i_1 & i_2 \cdots i_n \\ j_1 & j_2 \cdots j_n \end{pmatrix}.$$

We may relabel the subscripts k so that

$$H^* = \begin{pmatrix} 1 & 2 \cdots n \\ j_1 & j_2 \cdots j_n \end{pmatrix},$$

i.e. such that $i_k = k$.

If we decompose H into canonical cycles we have either

(i) $H^* = (k_1 k_2 \cdots k_n)$ or

(ii) $H^* = (k_1 k_2 \cdots k_m)(k_{m+1} k_{m+2} \cdots k_{m+m^*}) \cdots$ with $m < n$

where $k_1, k_2, \cdots, k_n$ is a rearrangement of $(1, 2, \cdots, n)$. In case we have (ii), let K be the subset of H consisting of the pairs (k_1k_2), (k_2k_3), $\cdots$, $(k_{m-1}k_m)(k_mk_1)$ and let L be the set of the remaining pairs of H. Then the pairs of L are of type (k_ik_j) with $i > m$ and $j > m$. Consider the point

$$P = x_{k_1} x_{k_2} \cdots x_{k_m} x_{k_{m+1}}' \cdots x_{k_n}'.$$

Then P is in $D(f)$. However,

$$P \notin D(x_{k_i} x_{k_j}') \text{ if } i < m \quad \text{and} \quad j < m,$$

and

$$P \notin D(x_{k_i} x_{k_j}') \text{ if } i > m \quad \text{and} \quad j > m,$$

and hence

$$P \notin D\left(\sum_{(uv) \in K} x_u x_v' \right) \quad \text{and} \quad P \notin D\left(\sum_{(uv) \in L} x_u x_v' \right).$$

This is a contradiction since

$$D\left(\sum_{(uv) \in K} x_u x_v' \right) \cup D\left(\sum_{(uv) \in L} x_u x_v' \right) = D\left(\sum_{(uv) \in H} x_u x_v' \right) = D(f).$$

Therefore case (ii) cannot occur, and hence we must have case (i), *i.e.*, H^* is an n-cycle.

Conversely given an arbitrary n-cycle $(q_1 q_2 \cdots q_n)$, in view of the complete symmetry in the subscripts $(1, 2, \cdots, n)$, Lemma 7 tells us that

$$x_{q_1} x_{q_2}' + x_{q_2} x_{q_3}' + \cdots + x_{q_{n-1}} x_{q_n}' + x_{q_n} x_{q_1}'$$

is in $Zsp(f)$.

If a and b are two distinct permutations of $1, 2, \cdots, n$, then for some i, $a(i) \neq b(i)$ so that

$$x_1 x_{a(1)}' + x_2 x_{a(2)}' + \cdots + x_n x_{a(n)}' \neq x_1 x_{b(1)}' + \cdots + x_n x_{b(n)}'.$$

Given an n-cycle $H^* = (k_1, k_2, \cdots, k_n)$ we may cyclically permute the k's so as to have $k_1 = 1$, and with this stipulation the n-cycle H^* will uniquely determine $k_2, k_3, \cdots, k_n$, and $(k_2, k_3, \cdots, k_n)$ is then a rearrangement of $(2, 3, \cdots, n)$. Conversely given a permutation

$$\begin{pmatrix} 2 & 3 \cdots n \\ k_2 & k_3 \cdots k_n \end{pmatrix}$$

of $(2, 3, \cdots, n)$ it uniquely determines an n-cycle $(1\ k_2\ k_3 \cdots k_n)$. Thus, there are $(n-1)!$ distinct n-cycles.

We summarize the results of this section in the following theorem.

Theorem 4: Consider the Boolean expression $f(x_1, \cdots, x_n) = (x_1 x_2 \cdots x_n + x_1' x_2' \cdots x_n')'$ with $n > 1$. There are $N = (n-1)!$ different permutations $a_1, a_2, \cdots, a_N$ of $1, 2, \cdots, n$ which are n-cycles. For the n-cycle a_i let

$$g_i(x_1, \cdots, x_n) = x_1 x_{a_i(1)}' + x_2 x_{a_i(2)}' + \cdots + x_n x_{a_i(n)}'.$$

Then $g_1, g_2, \cdots, g_N$ are all the members of $Zsp(f)$ and $LZsp(f) = 2n-1$. One way of enumerating the different n-cycles is to enumerate the $N = (n-1)!$ permutations

$$b_i = \begin{pmatrix} 2 & 3 & \cdots n \\ k_2 & k_3 \cdots & k_n \end{pmatrix} \qquad (i = 1, 2, \cdots, N)$$

of $2, 3, \cdots, n$, and let a_i be the n-cycle $(1\ k_2\ k_3 \cdots k_n)$, *i.e.*,

$$a_i = \begin{pmatrix} 1 & k_2 & k_3 \cdots k_{n-1} & k_n \\ k_2 & k_3 & \cdots \quad k_n & 1 \end{pmatrix},$$

so that

$$g_i(x_1, \cdots, x_n) = x_1 x_{k_2}' + x_{k_2} x_{k_3}' + x_{k_3} x_{k_4}' + \cdots + x_{k_{n-1}} x_{k_n}' + x_{k_n} x_1'.$$

Example: Take $n = 4$ so that

$$f(x_1, x_2, x_3, x_4) = (x_1x_2x_3x_4 + x_1'x_2'x_3'x_4')'.$$

There are $(4-1)! = 3! = 6$ permutations of 2, 3, 4. They are

$$b_1 = \begin{pmatrix} 2, & 3, & 4 \\ 2, & 3, & 4 \end{pmatrix}, \quad b_2 = \begin{pmatrix} 2, & 3, & 4 \\ 2, & 4, & 3 \end{pmatrix}, \quad b_3 = \begin{pmatrix} 2, & 3, & 4 \\ 3, & 2, & 4 \end{pmatrix},$$

$$b_4 = \begin{pmatrix} 2, & 3, & 4 \\ 3, & 4, & 2 \end{pmatrix}, \quad b_5 = \begin{pmatrix} 2, & 3, & 4 \\ 4, & 2, & 3 \end{pmatrix}, \quad b_6 = \begin{pmatrix} 2, & 3, & 4 \\ 4, & 3, & 2 \end{pmatrix}.$$

The corresponding 4-cycles are

$$a_1 = (1, 2, 3, 4), \quad a_2 = (1, 2, 4, 3), \quad a_3 = (1, 3, 2, 4),$$
$$a_4 = (1, 3, 4, 2), \quad a_5 = (1, 4, 2, 3), \quad a_6 = (1, 4, 3, 2).$$

There are 6 minimal *sp* expressions $g_1, g_2, g_3, g_4, g_5, g_6$ equivalent to f, their length is 7, and we can write them down, using the 4-cycles a_i, in the following two ways:

$$g_1(x_1, x_2, x_3, x_4) = x_1x_2' + x_2x_3' + x_3x_4' + x_4x_1'$$
$$= x_1x_2' + x_2x_3' + x_3x_4' + x_4x_1'.$$
$$g_2(x_1, x_2, x_3, x_4) = x_1x_2' + x_2x_4' + x_4x_3' + x_3x_1'$$
$$= x_1x_2' + x_2x_4' + x_3x_1' + x_4x_3'.$$
$$g_3(x_1, x_2, x_3, x_4) = x_1x_3' + x_3x_2' + x_2x_4' + x_4x_1'$$
$$= x_1x_3' + x_2x_4' + x_3x_2' + x_4x_1'.$$
$$g_4(x_1, x_2, x_3, x_4) = x_1x_3' + x_3x_4' + x_4x_2' + x_2x_1'$$
$$= x_1x_3' + x_2x_1' + x_3x_4' + x_4x_2'.$$
$$g_5(x_1, x_2, x_3, x_4) = x_1x_4' + x_4x_2' + x_2x_3' + x_3x_1'$$
$$= x_1x_4' + x_2x_3' + x_3x_1' + x_4x_2'.$$
$$g_6(x_1, x_2, x_3, x_4) = x_1x_4' + x_4x_3' + x_3x_2' + x_2x_1'$$
$$= x_1x_4' + x_2x_1' + x_3x_2' + x_4x_3'.$$

Section 9

Theorem 5: Consider the Boolean expression

$$f(x_1, \cdots, x_n) = x_1x_2 \cdots x_n + x_1'x_2' \cdots x_m'x_{m+}x_{m+2} \cdots x_n$$

with $1 \leqq m \leqq n$. Let $a_1, a_2, \cdots, a_M$ be the $M=(m-1)!$ permutations of 1, 2, $\cdots$, m which are m-cycles. Let $k_i = a_k(i)$ and let

$$h_i(x_1, \cdots, x_n) = (x_1' + x_{k_1})(x_2' + x_{k_2}) \cdots (x_m' + x_{k_m})x_{m+}x_{m+2} \cdots x_n.$$

Then $h_1, h_2, \cdots, h_M$ are all the members of $Zps(f)$ and $LZps(f) = n+m-1$.

Proof: Let

$$F(x_1, x_2, \cdots, x_m) = x_1x_2 \cdots x_m + x_1'x_2' \cdots x_m'.$$

By Theorem 4, we know that for

$$F'(x_1, x_2, \cdots, x_m) = (x_1x_2 \cdots x_m + x_1'x_2' \cdots x_m')'$$

the members of $Zsp(F')$ are

$$\sum_{i=1}^{m} x_ix_{k_i}'; \qquad k = 1, 2, \cdots, M = (m-1)!.$$

Dualizing this we conclude that the members of $Zps(F)$ are

$$\prod_{i=1}^{m}(x_i' + x_{k_i}); \qquad k = 1, 2, \cdots, M.$$

Now there are only two points $x_1x_2 \cdots x_n$ and $x_1'x_2' \cdots x_m'x_{m+1}x_{m+2} \cdots x_n$ in the point-complex of $f(x_1, x_2, \cdots, x_n)$ in the n-cube S_n, and these two points are contained in the m-cube S_m given by

$$H(x_1, x_2, \cdots, x_n) = x_{m+1}x_{m+2} \cdots x_n.$$

The point-complex of $F(x_1, x_2, \cdots, x_m)$ in this m-cube S_m is the same as the point-complex of $f(x_1, x_2, \cdots, x_n)$ in S_n. Therefore, by Theorem 1, the members of $Zps(f(x_1, x_2, \cdots, x_n))$ are obtained by multiplying each of the above M members of $Zps(F(x_1, x_2, \cdots, x_m))$ by $H(x_1, x_2, \cdots, x_n)$; that the length of each of these products is $n+m-1$ is clear; and this proves the theorem.

Example: Take $m=4$ and $n=6$ so that

$$f(x_1, x_2, x_3, x_4, x_5, x_6) = x_1x_2x_3x_4x_5x_6 + x_1'x_2'x_3'x_4'x_5x_6.$$

Then using the previous example, we have the following $(4-1)!=6$ minimal ps expressions equivalent to f, and each is of length $6+4-1=9$.

$$h_1(x_1, x_2, x_3, x_4, x_5, x_6) = (x_1' + x_2)(x_2' + x_3)(x_3' + x_4)(x_4' + x_1)x_5x_6.$$
$$h_2(x_1, x_2, x_3, x_4, x_5, x_6) = (x_1' + x_2)(x_2' + x_4)(x_3' + x_1)(x_4' + x_3)x_5x_6.$$
$$h_3(x_1, x_2, x_3, x_4, x_5, x_6) = (x_1' + x_3)(x_2' + x_4)(x_3' + x_2)(x_4' + x_1)x_5x_6.$$
$$h_4(x_1, x_2, x_3, x_4, x_5, x_6) = (x_1' + x_3)(x_2' + x_1)(x_3' + x_4)(x_4' + x_2)x_5x_6.$$
$$h_5(x_1, x_2, x_3, x_4, x_5, x_6) = (x_1' + x_4)(x_2' + x_3)(x_3' + x_1)(x_4' + x_2)x_5x_6.$$
$$h_6(x_1, x_2, x_3, x_4, x_5, x_6) = (x_1' + x_4)(x_2' + x_1)(x_3' + x_2)(x_4' + x_3)x_5x_6.$$

Theorem 6: Let the notation be as in Theorem 5. Then $LZsps(f) = LZps(f) = n+m-1$ so that $Zps(f) \subset Zsps(f)$. If $m<n$ then $Zps(f) = Zsps(f)$ so that $h_1, h_2, \cdots, h_M$ are all the members of $Zsps(f)$, and if $m=n$ then f, h_1, h_2, $\cdots$, h_M are all the members of $Zsps(f)$.

Proof: Since $D(f)$ consists of two points (say $P_1 = x_1x_2 \cdots x_n$ and $P_2 = x_1'x_2' \cdots x_m'x_{m+1} \cdots x_n$), the considerations of Section 5 tell us that as possible candidates for $Zsps(f)$ we haee to consider the following two cases

(i) $h \sim f$ with h of type product-sum;

(ii) $h = q_1 + q_2 \sim f$ with q_1 and q_2 of type product-sum, $D(q_1) = P_1$ and $D(q_2) = P_2$.

By Theorem 5, the minimal length for h of type (i) is $n+m-1$. In case (ii) Theorem 3 tells us that

$$L(h) = L(q_1) + L(q_2) + 1 = (n-1) + (n-1) + 1 = n + n - 1.$$

Therefore,

$$LZsps(f) = \min(n+m-1, n+n-1) = n+m-1 = LZps(f).$$

Hence $Zps(f) < Zsps(f)$, *i.e.*, $h_1, h_2, \cdots, h_M$ are all the distinct members of $Zsps(f)$ which arise from case (i). Now case (ii) can arise if, and only if, $n+m-1 = n+n-1$, *i.e.*, $n=m$, and in that case Theorem 3 tells us $q_1 = x_1x_2 \cdots x_n$ and $q_2 = x_1'x_2' \cdots x_m'x_{m+1}x_{m+2} \cdots x_n$ so that $h=f$.

Example: First take $m=4$ and $n=6$ so that

$$f(x_1, x_2, x_3, x_4, x_5, x_6) = x_1x_2x_3x_4x_5x_6 + x_1'x_2'x_3'x_4'x_5x_6.$$

Then the $(4-1)!=6$ expressions $h_1, h_2, h_3, h_4, h_5, h_6$ given in the previous example are exactly the minimal sum of product of sums expressions equivalent to f and their length, *i.e.*, $LZsps(f)=g$.

Second, take $m=n=4$ so that

$$f(x_1, x_2, x_3, x_4) = x_1x_2x_3x_4 + x_1'x_2'x_3'x_4'.$$

Then there are *seven* minimal sum of products of sums expressions equivalent to f, and the length of each of them is $4+4-1=7$, namely:

$$f(x_1, x_2, x_3, x_4) = x_1x_2x_3x_4 + x_1'x_2'x_3'x_4'.$$

$$h_1(x_1, x_2, x_3, x_4) = (x_1' + x_2)(x_2' + x_3)(x_3' + x_4)(x_4' + x_1).$$

$$h_2(x_1, x_2, x_3, x_4) = (x_1' + x_2)(x_2' + x_4)(x_3' + x_1)(x_4' + x_3).$$

$$h_3(x_1, x_2, x_3, x_4) = (x_1' + x_3)(x_2' + x_4)(x_3' + x_2)(x_4' + x_1)$$

$$h_4(x_1, x_2, x_3, x_4) = (x_1' + x_3)(x_2' + x_1)(x_3' + x_4)(x_4' + x_2).$$

$$h_5(x_1, x_2, x_3, x_4) = (x_1' + x_4)(x_2' + x_3)(x_3' + x_1)(x_4' + x_2).$$

$$h_6(x_1, x_2, x_3, x_4) = (x_1' + x_4)(x_2' + x_1)(x_3' + x_2)(x_4' + x_3).$$

Section 10

Let $f(x_1, \cdots, x_n)$ be a Boolean expression such that $D(f)$ consists of two points P_1 and P_2. We want to find $Zsps(f)$. If P_1 and P_2 form a 1-cell then $Zps(f)$ is given in Theorem 2, and as will be shown in our forthcoming paper,[6] $Zsps(f)=Zps(f)$. Now assume P_1 and P_2 do not lie on a 1-cell, *i.e.*, that P_1 and P_2 are the only cells in $C(f)$. Let S_m be the m-cell spanned by P_1 and P_2; then $m>1$. Let $(k_1, k_2, \cdots, k_m)=K(S_m)$, $(i_1, i_2, \cdots, i_s)=I(S_m)$ and $(j_1, j_2, \cdots, j_t)=J(S_m)$ so that $s+t=n-m$. Let

$$y_q = x_{k_q} \quad \text{for } q = 1, 2, \cdots, m;$$
$$y_{m+q} = x_{i_q} \quad \text{for } q = 1, 2, \cdots, s; \text{ and}$$
$$y_{m+s+q} = x_{j_q}' \quad \text{for } q = 1, 2, \cdots, t.$$

Then

$$f(x_1, \cdots, x_n) \sim x_{k_1}x_{k_2}\cdots x_{k_m}x_{i_1}\cdots x_{i_s}x_{j_1}'\cdots x_{j_t}'$$
$$+ x_{k_1}'x_{k_2}'\cdots x_{k_m}'x_{i_1}x_{i_2}\cdots x_{i_s}x_{j_1}'x_{j_2}'\cdots x_{j_t}'$$
$$= y_1y_2\cdots y_n + y_1'y_2'\cdots y_m'y_{m+1}y_{m+2}\cdots y_n,$$

and we are thus reduced to Theorem 6.

Example: First, take $n=3$ and suppose that $f(x_1, x_2, x_3) = x_1x_2'x_3' + x_1x_2'x_3$. Then both the "points" $x_1x_2'x_3'$ and $x_1x_2'x_3$ lie on the 1-cell x_1x_2', and hence f itself is the only minimal *sps* expression equivalent to itself.

Second, take $n=6$ and suppose that

$$f(x_1, x_2, x_3, x_4, x_5, x_6) = x_2x_5x_3x_6x_4x_1 + x_2'x_3x_6'x_4'x_1x_5'.$$

Set

$$y_1 = x_2,\ y_2 = x_5,\ y_3 = x_6,\ y_4 = x_4,\ y_5 = x_3,\ y_6 = x_1.$$

Then

$$f(x_1, x_2, x_3, x_4, x_5, x_6) = f^*(y_1, y_2, y_3, y_4, y_5, y_6)$$
$$= y_1y_2y_3y_4y_5y_6 + y_1'y_2'y_3'y_4'y_5y_6.$$

Hence, we can apply the first part of the previous example with y_i in place of x_i. In other words, we can apply that example with the following *relabelling* of the x_i: $x_1 \to x_6$, $x_2 \to x_1$, $x_3 \to x_5$, $x_4 \to x_4$, $x_5 \to x_2$, $x_6 \to x_3$.

Third, take $n=4$ and suppose that

$$f(x_1, x_2, x_3, x_4) = x_3x_1x_4x_2 + x_3'x_4'x_2'x_1'.$$

Then

$$f(x_1, x_2, x_3, x_4) = x_1x_2x_3x_4 + x_1'x_2'x_3'x_4'$$

and this is just the second part of the previous example.

A Method for Synthesizing the Waveform Generated by a Character, Printed in Magnetic Ink, in Passing Beneath a Magnetic Reading Head*

I. FLORES† AND F. RAGONESE†

Summary—When a character printed in magnetic ink passes beneath a magnetic reading head it generates a waveform. A method is described here to determine this waveform from the geometry of the printed character. The character shape is divided into elementary vertical units and the height of these units is then tabulated. A single set of experimental data is obtained in the laboratory by passing a magnetically printed bar beneath the same reading head which will be used to read the characters. Formulas are derived in this paper for combining the geometrical data obtained from the character with the experimental data obtained from the laboratory run with the head. A method is then described for programing a high-speed digital computer to derive the waveforms. The discussion explains why these waveforms are often superior to what might be obtained in an actual laboratory run of the printed characters.

Introduction

THE purpose of this article is to describe the theoretical aspects and the computational problems involved in synthesizing waveforms generated from magnetic ink printing which passes beneath a magnetic reading head.

This problem arises from investigations in character recognition. Printing characters in magnetic ink for automatic reading and recognition during document processing is attractive from several points of view. Magnetic printing is highly resistant to most kinds of mutilation. The application of fountain pens, ball point pens, pencils, stamp pad, ink, dirt, or grease to the surface of magnetic ink printing has practically no effect upon the waveforms it generates. Also, folding, bending, crumpling or otherwise abusing the document has little effect upon the waveform generated by the magnetically printed character. Furthermore, the methods for printing such documents can be those commonly employed in the printing field.

Experimental waveforms derived from magnetic printing are not determined solely by the originally designed geometry of the characters. These waveforms are affected by the chemical composition of the ink, the squeeze-out around the impression, the fibre structure of the paper, the absorbency of the paper and other similar factors. These factors produce a three-dimensional configuration substantially similar to the original design but with contaminations which produce aberrations in the waveform. To facilitate design and to study individually the effect of the variables of printing, it is desirable to know the waveform generated by an "ideal" printed character free from the contaminations of printing, paper and inks. This article describes a procedure for deriving these waveforms by using digital computer techniques, and also presents the rationale for this procedure.

Such a synthesizing procedure also enables the designer to alter the characters to qualitatively change the waveforms generated by them in a predictable manner.

Analysis of the Problem

It is required to find the voltage induced in the coil of a reading head while a character printed in magnetic ink and previously magnetized passes beneath the head. This voltage is directly proportional to the rate of change of the magnetic flux through the coil of the reading head. The total flux of the core material passes through the coil. The flux in the core is determined by a number of factors, the most important of which are the geometry of the core and the characteristics of the material from which it is made, especially its permeance. Thus, as a fixed magnetic pattern is placed beneath the core, the flux passing through the core will be determined by the factors just mentioned—its geometry and physical characteristics.

The pattern surrounding a magnetic geometric form is not easily represented in an analytic equation. In the very simple case of a magnetized rectangular area, for example, the flux at any point in the immediate vicinity surrounding this rectangle (even assuming point magnetic sources) is represented in terms of a vector sum of two quadratic terms.

To start with a given geometric form and to determine analytically the voltage induced in the coil appears to be a rather difficult problem. The first step, to find the flux concentration around the geometric form, is a task in itself. In addition, as mentioned previously, the flux picked up by the core depends on the geometry and physical characteristics of the core.

The approach here is, therefore, to start with empirical data. Such data would be in the form of waveshapes experimentally recorded in the laboratory when some basic unit of area passes beneath the reading head. It is next necessary to decide what basic unit will provide the best basis for a computational approach.

The waveshape produced by a rectangle or bar passing beneath a reading head is well known. It consists of a positive pulse generated by the leading edge of the bar and a negative pulse generated by the trailing edge of

* Manuscript received by the PGEC, February 27, 1958; revised manuscript received, June 19, 1958.
† Remington Rand Univac, South Norwalk, Conn.

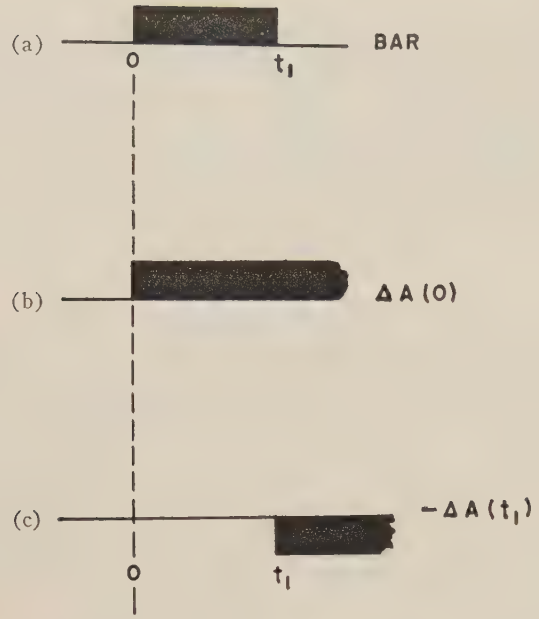


Fig. 1—(a) Printed bar. (b), (c) Two printed steps, $\Delta A(0)$ and $-\Delta A(t_1)$, which together make up equivalent response as bar (a).

the bar. (The direction is of course arbitrary.) Such a bar might comprise a fundamental unit. It was felt, however, that a step would provide a better approach. The bar can be simulated by the sum of a positive and a negative step, as illustrated in Fig. 1.

It can be easily seen that as the step approaches the reading head, more and more flux lines will pass through the core. The rate of change of these flux lines will become maximum when the leading edge of the bar enters beneath the gap. As the full area of the bar passes under the gap, the flux concentration becomes maximum. The rate of change remains zero while the full area of the bar is under the gap. What is generated by a step is therefore a positive pulse which is centered at the leading edge of the step. The unit differential of the area will be the basic unit for finding a waveform of the character. The step and the voltage it generates are shown in Fig. 2. Such a step generates a unit pulse which is indicated by

$$\Delta A(0) \rightarrow e(t). \tag{1}$$

A multiple of the unit step will generate a pulse which is a multiple of the pulse generated by a unit step. This might be indicated as

$$k\Delta A(0) \rightarrow ke(t) \tag{2}$$

where $\Delta A(0)$ indicates the unit differential of area and (0) indicates that the leading edge of the step passes beneath the gap at time zero. The arrow is used here to indicate that the area on the left-hand side generates the voltage on the right-hand side. This is not an equation, but rather a device used by mathematicians to indicate the mapping correspondence between two classes of functions.

The time relationship, which states that a unit step passed beneath the head at a later time than that of

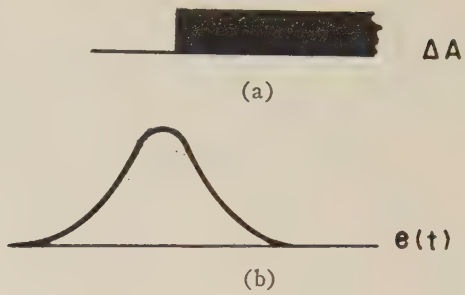


Fig. 2—(a) Unit step, $\Delta A(0)$. (b) Voltage, $e(t)$, generated by unit step.

$\Delta A(0)$ will generate a pulse delayed by the same amount is indicated by

$$\Delta A(\tau) \rightarrow e(t - \tau). \tag{3}$$

Eqs. (2) and (3) are combined as

$$k\Delta A(\tau) \rightarrow ke(t - \tau). \tag{4}$$

The final necessary relationship states that two steps of different size and occurring at different times produce voltages which are linear and additive; it is expressed by

$$k_1\Delta A(t_1) + k_2\Delta A(t_2) \rightarrow k_1e(t - t_1) + k_2e(t - t_2). \tag{5}$$

This last expression indicates that the three factors—proportionality, a linear time relationship and an additive voltage relationship—are all necessary for our computational process. These factors can be easily derived from the principle of superposition and the two equations

$$\phi = F/R \tag{6}$$

and

$$E = k\frac{d\phi}{dt}. \tag{7}$$

Procedure

The character in which we are interested consists of one or more bounding curves and is a connected closed area. The important aspect of this area is its direction of travel with respect to the reading head. This will be called the t axis. Such a form is shown in Fig. 3(a). Its area, integrated normal to the t axis, is shown in Fig. 3(b). The units to be used here must correspond to the definition of the unit step. This area is now divided along the t axis into slices of length Δt. Δt will correspond to that used later in reference to the voltage curve $e(t)$. The number of differentials of area in the ith slice A_i is now found for integral multiples of Δt. A plot of such values of the A_i's is shown in Fig. 3(b).

The difference in area between one slice and the rest will generate a voltage pulse. The amplitude of this pulse is proportional to the difference in area between the two slices. The difference between the ith slice and the $(i-1)$th slice may be called

$$\delta_i = A_i - A_{i-1}. \tag{8}$$

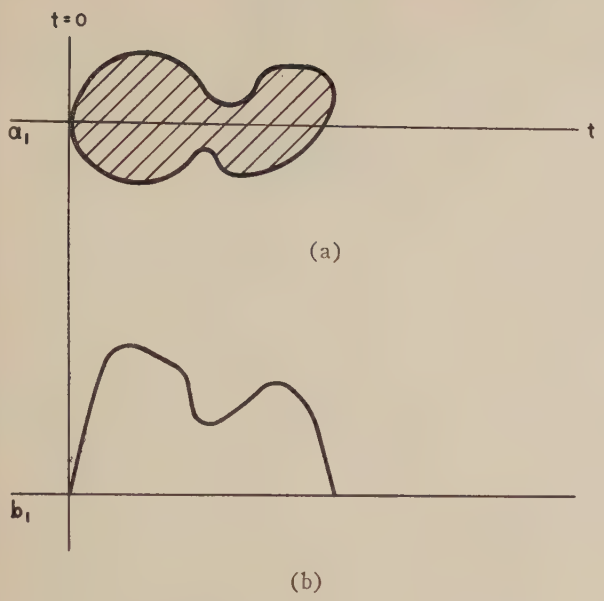


Fig. 3—(a) Printed pattern. (b) Plot of area of (a).

Because of the properties of linearity, proportionality and additivity, these relationships may be superimposed.

The data taken in the laboratory for the voltage derived from the printed step are now "calibrated," using the same value for Δt as above. The amplitude is plotted against time as in Fig. 2 and the maximum is reached at time $t=0$. This "calibration" yields a series of significant values of $e(t)$ with

$$e_k = e(k\Delta t) \tag{9}$$

where k can vary from $-L$ to $+M$; e_k is approximately zero for values of k less than $-L$ and greater than $+M$.

The first area slice or step area generates a voltage which may be called $E_1(t)$ and which will be equal to $\delta_1 e(t-\Delta t)$. It is illustrated in Fig. 4(c) for the area shown in Fig. 4(a) whose differentials are in Fig. 4(b). At the time $j\Delta t$ the voltage $E_1(j\Delta t)$ will be called E_{ij}. Then

$$E_{11} = \delta_1 \cdot e_0 \tag{10}$$

and

$$E_{12} = \delta_1 \cdot e_1 \qquad E_{10} = \delta_1 e_{-1} \tag{11}$$

or

$$E_{1j} = \delta_1 \cdot e_{j-1} \tag{12}$$

where

$$-L + 1 \leq j \leq M + 1 \tag{13}$$

and

$$E_{1j} = 0 \text{ for } j < -L + 1, j > M + 1. \tag{14}$$

$E_2(t)$ and $E_3(t)$ are similarly defined and illustrated in Fig. 4(a) and 4(e). The general term $E_k(t)$ is defined in the same manner:

$$E_k(t) = \delta_k e(t - k\Delta t). \tag{15}$$

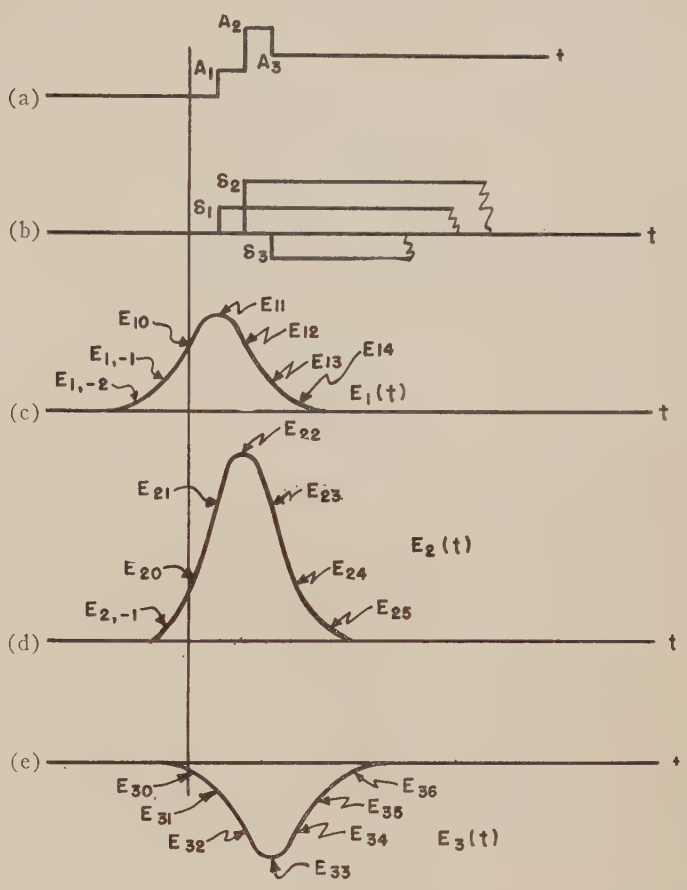


Fig. 4—(a) Area for waveform. (b) Differentials of area. (c) Resultant voltage for δ_1. (d) Resultant voltage for δ_2. (e) Resultant voltage for δ_3.

Define

$$E_k(j\Delta t) = E_{kj} \tag{16}$$

so that

$$E_{kk} = \delta_k \cdot e_0 \tag{17}$$

and

$$E_{k,k+1} = \delta_k \cdot e_1, \qquad E_{k,k-1} = \delta_k e_{-1} \tag{18}$$

or

$$E_{kj} = \delta_k e_{j-k} \tag{19}$$

where

$$-L + k \leq j \leq M + k \tag{20}$$

and

$$e_{kj} = 0 \text{ for } j < -L + k, j > M + k. \tag{21}$$

The output voltage E_T from the coil, which is to be synthesized, is found by adding together all the voltages derived at one given time from all the steps which create an output at that time. The total voltage produced at time j is given by

$$E_{Tj} = E_{1j} + E_{2j} + \cdots + E_{Nj} \tag{22}$$

so that

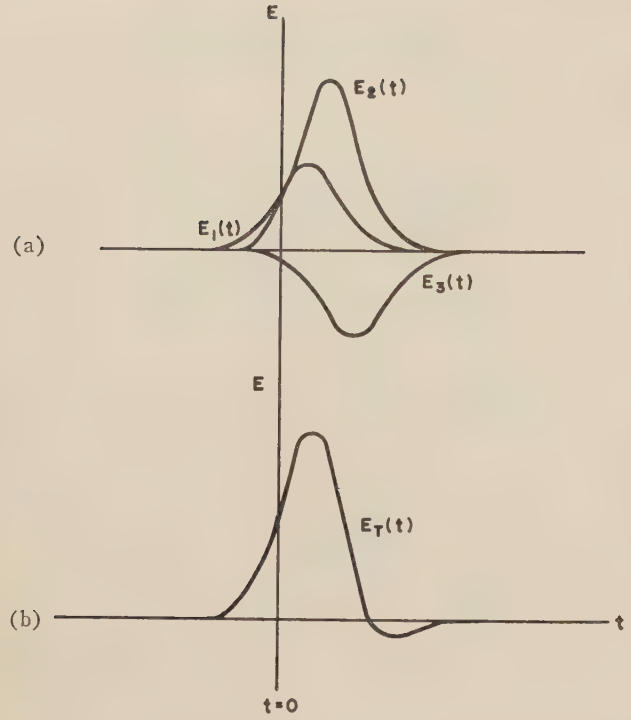


Fig. 5—(a) E_1, E_2, and E_3 of Fig. 4 superimposed. (b) Sum. E_T.

$$E_{T1} = E_{11} + E_{21} + \cdots + E_{L+1,1} \tag{23}$$

and

$$E_{T1} = \sum_{k=1}^{N} E_{k1} \text{ where } E_{k1} = 0 \text{ for } k > L + 1 \tag{24}$$

and in general

$$E_{Tj} = \sum_{k=1}^{N} E_{kj-L+1 \le j \le N+M} \tag{25}$$

where

$$E_{kj} = 0 \text{ for } k < j - M \quad \text{and} \quad k > j + L. \tag{26}$$

The curves for E_1, E_2 and E_3 for the area of Fig. 4(a) are shown superimposed on Fig. 5(a), and the result of adding these curves together is shown in Fig. 5(b).

Computer Calculation

The data to be supplied to the computer will be the $G+M+1$ constants e_{-L}, e_{-L+1}, $\cdots$, e_{-1}, eo, e_1, $\cdots$ e_{m-1}, e_m. These are the constants obtained from the experimental data. Also supplied to the computer as raw data are the δ_j's, which are obtained by the following method. An enlarged image of the character whose waveform is to be synthesized is drawn on quadrangle paper. The grid to be used would consist of lengths Δt in the horizontal direction and ΔA in the vertical direction. From this is prepared a chart giving the boxes for each slice ending at $k\Delta t$, which is found by simply counting the squares in the vertical column bounded by $(k-1)\Delta t$ and $k\Delta t$.

The computer then makes these calculations:

$$\delta_k = A_k - A_{k-1} \tag{27}$$

$$E_{kj} = \delta_k e_{j-k} \tag{28}$$

$$E_{Tj} = \sum_{k=1}^{N} E_{kj} \tag{29}$$

where

$$E_{kj} = 0 \text{ for } k > j + L \quad \text{and} \quad k < j - M \tag{30}$$

and

$$-L + 1 \le j \le N + M. \tag{31}$$

The calculations required make use of subtraction for (27), multiplication for (28) and a series of additions for (29). These calculations can be simply programed on any modern computer.

The result of these computations can then be graphed on the same time scale in units of Δt. Notice that this output voltage has values which range from $-L+1$ to $N+M$. This is because the voltage in the coil begins before the printed form reaches the head gap and remains after the form has passed beneath the reading head gap.

Illustrative Example

As an example of how this method is used, the waveform for the printed character "0" has been synthesized. The character itself is shown in Fig. 6(a). In Fig. 6(b) the area is plotted along the t axis; each of the vertical values is one of the A_k's. The increments of area designated as δ_k appear in Fig. 6(c). Although these δ_k's are plotted, they are not necessary for the computations and are not supplied in the form of raw data. The result of the maximum computations appears in Fig. 6(d). Notice that this output voltage consists of a positive and negative pulse followed a little later by another positive and negative pulse. These pulses correspond to the large increments in area. The first pulse is caused by the left-hand edge of the zero; the first negative pulse is caused by the right-hand trailing edge of the left hand bar of the zero; second positive pulse is caused by the left-hand edge of the right-hand bar of the zero; the second negative pulse is caused by the right-hand edge of the character. It is generally true that the waveshape generated by any character printed in magnetic ink will be determined by the larger increments in an area.

Although the two characters "0" and "8" appear to be quite different as shown in Figs. 7(a) and 7(b), the waveforms generated by these two characters are almost identical. An automatic character-reading system could not use the characters whose generated waveforms correspond to such a degree. This difficulty could be overcome by using a different figure "8," as shown in Fig. 7(c). This character would generate four sets of positive and negative pulses instead of the two sets of positive and negative pulses which would be generated by the character in Fig. 7(b).

Through the study of printed characters and the waveforms which they generate, character designs can

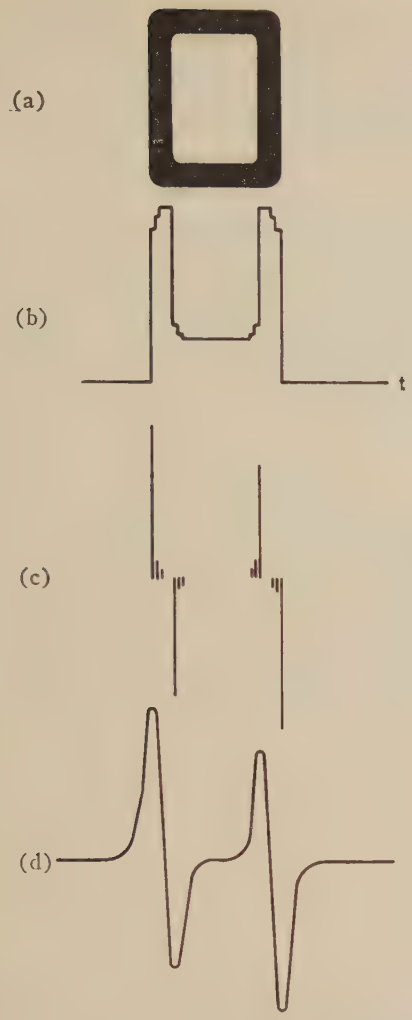


Fig. 6—(a) Character "0." (b) Vertical components, A_k. (c) k for "0." (d) $E_T(t)$ for "0."

Fig. 7—(a) Character "0." (b) Character "8," same waveform as (a). (c) Character "8," waveform different from (a).

be made which are as easily distinguishable from the waveforms that they generate in magnetic reading as they are distinguishable in human reading. This is one of the needs that the synthesis approach fulfills.

Discussion of Results

The results obtained from such an automatically programed machine computation agree very closely with the waveforms taken in practice in the laboratory from printed samples. The question arises, why not simply take the waveforms in the laboratory? The reason for not using the laboratory samples is that printing introduces certain distortions which cannot be closely controlled. To average the results taken from a given batch of printing would not be correct, since some of the distortions are due to long term parameters. Such parameters would be present in a given batch and would change only from one batch to another. An example of such parameters would be the type of impression used by the printer or the amount of ink in the fountain of the printing press.

The theoretical results use printed samples to derive only the function $e(t)$. A square of magnetized metal may give a more exact measure of $e(t)$. This function may be studied and also smoothed to reduce the eccentricities of the printing process, if printing is used to derive it. The waveforms predicted by using such a smoothed function, $e(t)$, will be more suitable theoretically than those derived by averaging the laboratory waveforms.

Somewhat better results are obtained when the edge phenomenon is taken into account. Although the edge of the bar under consideration may be designed to print as a straight line, it looks more like a topographical map of the Rocky Mountains when viewed under a microscope. The ink is transferred to the paper from the printing roll. The way the ink prints on the paper depends on the roller pressure, the amount of ink on the roller, the viscosity of the ink, the absorbency of the paper, the fibre structure of the paper, etc. The ink distribution at the edge which is designed to print as a vertical line may be approximated by a probability density distribution with very good results. The distances being considered are in the neighborhood of fractions of a thousandth of an inch. This is well below the resolution of the head. The probability density distribution which was arrived at from physical considerations

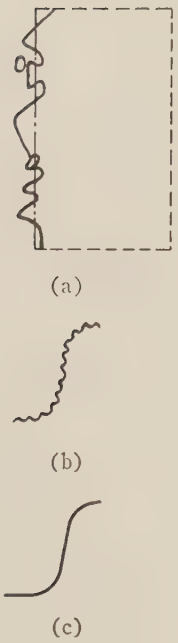


Fig. 8—(a) Ink density at edge. (b) Vertically integrated ink density. (c) Probability density approximation to vertically integrated ink density.

is very close to the actual flux distribution prevailing in the head. Fig. 8(a) illustrates the edge effect described, (b) the vertically integrated area at the edge, and (c) the probability density distribution.

Conclusion

A method has been discussed for deriving the waveforms generated as a character printed in magnetic ink is passed underneath a magnetic reading head. The method employs the geometry of the character and the empirical properties of the magnetic reading head, the ink, the printing process, etc., involved.

Bibliography

As far as the present authors know, nothing has been published regarding the reading of magnetically printed documents. For background in general magnetic recording, the following work is adequate:

S. J. Begun, "Magnetic Recording," Murray Hill Books, New York, N.Y.; 1949.

For a thorough treatment about computer applications of magnetic recording and a good bibliography of the subject see:

R. K. Richards, "Digital Computer Components and Circuits," Von Nostrand and Co., Princeton, N.J., pp. 314–353; 1957.

On the Minimum Logical Complexity Required for a General Purpose Computer*

S. P. FRANKEL†

Summary—**A definition is provided for the term "general purpose computer" (gpc) which is compatible with usage and is analogous to, but distinct from, Turing's definition of a "universal computer." A gpc is presented in functional and logical design which seems to approximate the minimum complexity consistent with this definition. No specific definition of complexity nor a bound on required complexity is presented.**

Definitions

THE term *general purpose computer* (gpc) has been applied to a variety of digital computing instruments having the following characteristics: 1) The ability to perform various arithmetic operations on numbers in digital form. These operations usually include addition, subtraction, multiplication, and division. Numbers are usually held in decimal or binary expansion to a convenient precision (number of decimal or binary digits) and are usually signed. That is, the arithmetic operations are performed on algebraic numbers expressed to limited precision. 2) The instrument includes a *memory store*, which holds both the numbers taking part in a calculation and the *instructions*, which control the execution of elementary operations. The sequence in which instructions held in the memory take control of activity is dependent (in some situations, at least) on the numbers produced by arithmetic operations. One familiar form of this *branching* is the selection of one or another subsequent chain of instructions depending on the sign of a number appearing in the calculation. 3) Numbers produced in the course of the calculation may be stored in the memory and thus made available for use at later stages of the calculation.

The appellation "general purpose" implies that by suitable choice of the numbers and instructions initially set in the memory, the computer may be enabled to carry out any one of a broad class of calculations. A

* Manuscript received by the PGEC, May 22, 1958.

† Consultant, Logical Design. 1764 Redondo Ave., Long Beach, Calif.

uantitative limitation on the breadth of this class is mposed by the capacity of the memory, *i.e.*, by the mount of instructional and numerical information it s able to hold. To remove this uninteresting limitation he term *general purpose* will be used here in a strict ense requiring that the memory capacity be arbitrarily xtensible without change in the logical design. By add- ng the clearly reasonable restriction that the informa- ion initially set in the memory (the *program*) be of nite extent, the present definition is made similar to he Turing definition of a *computer*. If, moreover, the set f elementary operations is of ordinary flexibility, the eneral purpose computer has the essential property of Turing's *universal computer*,[1] the ability to perform any omputation which could be done by any computer. It s the object of the present paper to display a gpc logical lesign of nearly minimal complexity compatible with his practical approximation to the requirements for a niversal computer.

At each step of its operation the *computer proper* oc- upies one of a finite number of *active states*. The term *omputer proper* describes those parts of the computer ubject to immediate change; as opposed to the memory, most parts of which are quiescent in each step of the omputation. One measure of complexity considered ere is the logarithm (to base two) of this number of ctive states, *i.e.*, the number of binary digits required o specify the condition of the computer proper.

The computer described below is of the *synchronous* ype. It makes use of a *clock* which presents timing ulses by means of which all activities are synchronized. The time interval between successive clock pulses is alled a *digit period*. In each digit period the computer roper receives from the memory a body of information which is typically very small compared with the full memory capacity. This information is received as a dis- crete set of binary digits (*bits*). Similarly, a few bits of nformation are deposited (*recorded*) in the memory in ach digit period. These recorded bits are functions, as described in the logical design, of the current active state and of the bits received from memory. At the time of each clock pulse the active state may change. The new active state is also a function, described in the logi- cal design, of the prior active state and of the bits re- ceived from the memory during the past digit period.

Various constituents of the complexity of a logical de- sign are: 1) the number of bits specifying the active state; 2) the number (or mean number) of bits received from memory in each digit period; 3) the number of bits re- corded in the memory in each digit period; 4) the com- plexity of the functions determining the bits recorded and the new state. These constituents may be combined (and the fourth given a quantitative measure) to pro- vide an over-all measure of complexity. Since no formal minimization of complexity is claimed for the design presented below, a choice of such an over-all measure need not be made here.

[1] A. M. Turing, "On computable numbers," *Proc. London Math. Soc.*, series 2, vol. 42 (1936–1937), pp. 230–265.

Functional Description of M'AC

The computer described here is designated *M'AC* (from MicrocephalAC). Its memory organ is a magnetic tape which presents and receives information in several channels. Each of two channels is served by two *heads* spaced by integral multiples of the distance corresponding to a digit period. One of these channels acts as a circulating register: the first of its heads records a bit in each digit period, the second (in the direction of tape motion) presents to the computer proper the bit which was recorded in a correspondingly earlier digit period. The other of these channels holds the main memory. The distance separating its two heads is an integral multiple of the distance between the circulating register heads. In some digit periods both heads read from the main memory; in others one (the memory writing head) records while the other reads. To avoid the necessity of reversing the direction of tape motion, with consequent disturbance of the storage of information in the circulating register, the tape is given the form of a loop.

Since the tape loop is of finite size the memory capacity is finite; thus, *M'AC* is not a *computer* in the sense of Turing. However, the loop size need not be fixed by the logical design but may be treated as a parameter available for choice in the planning of each computation. (In the same way the memory capacity of a typical gpc might be left unspecified in its logical design, thus justifying the use of the term "general purpose" in the stricter sense described above.)

In addition to the two channels described above, the tape may hold permanently recorded signals used to generate the clock pulse and two bits of timing information. Successive groups of n digit periods are called "word periods." The last digit period of each word period is distinguished by a signal, denoted L. The absence of the signal L is denoted $\overline{L}$ and marks bit periods other than the last of a word period. Another signal, N, and its complement, $\overline{N}$, distinguish alternate word periods. Word signals in which N is present are called *number word periods*, those in which $\overline{N}$ occurs are *instruction word periods*. The mechanism by which the signals N and $\overline{N}$ are generated may consist of a flip flop operated as a scale-of-two counter, changing its state at the end of each digit period in which L occurs. For reasons described below the signal N, unlike L, is not presented directly by the tape.

The distance through which the tape moves in a word period will be called a *word length*. The separation between the two heads of the circulating register is two word lengths, so that if the bit put on the tape by the recording head (denoted R'') were always that being presented by the reading head (denoted R), then an arbitrary sequence of $2n$ bits would be recirculated indefinitely. In fact, the equality $R''=R$ holds often but not invariably, in keeping with customary usage of the term

circulating register. Similarly, the separation of the two main memory heads is an integral multiple of two word lengths so that a word recorded by the recording head in a number word period will be read by the reading head in another number word period, several word periods later. The number of bits in a word, n, is not fixed by the logical design but, like the length of the tape loop, may be treated as a parameter available to the user's choice.

The $2n$ bits stored in the circulating register constitute two words: that presented (or recorded) during the N word period is a binary number resulting from previous computation and held for use in the next arithmetic operation; the bits presented or recorded in the $\overline{N}$ word period are related to the instruction awaiting execution. Thus, the circulating register plays the roles of both the accumulator and instruction circulating register of a simple binary digital computer. Similarly, information read from or recorded into the main memory is divided by the signals N and $\overline{N}$ into two categories: bits read during an $\overline{N}$ word period will be regarded as forming an instruction; those read or recorded during an N word period (recording into the main memory occurs only then) will be called a number. This distinction would appear to separate the main memory into disjoint halves; an instruction memory which can only be read and a numerical memory which can be recorded into or read. Such a separation would uncomfortably restrict the programmer's abilities. Accordingly, it is avoided by imposing the requirement that the length of the tape loop be an odd number of word lengths. Then each word of the main memory appears in successive tape revolutions in the guise of a number and an instruction alternately.

Two *phases* of operation are distinguished by a flip flop, denoted E. Its on state marks the *execute phase* in which an instruction is executed and a new instruction then read. An execute phase typically consists of a number word period followed by an instruction word period. Following this the *delay phase* marked $\overline{E}$ may be entered to await the appropriate time for the execution of the newly read instruction. Since the memory of $M'AC$ is entirely serial, an *address* consists only of the specification of the time of execution. It is presented by the main memory as a *delay number*, an integer describing the duration of the delay phase to follow in units of two word periods. (If the delay number is zero no delay phase is entered and the execute phase is continued for another pair of word periods.)

During the instruction word period of the execute phase, *i.e.*, in the circumstance $\overline{N}E$, a new instruction is set into the circulating register. It is derived from the bits presented by one of the two main memory heads, denoted M and M^* respectively. In subsequent instruction word periods of the delay phase the delay number is reduced by one with each circulation until it becomes negative, *i.e.*, until a one appears in its most significant bit position. At that time the execute phase is entered. (If the delay number as read from main memory was negative this condition arises immediately, hence no delay phase intervenes.)

The elementary operations of $M'AC$ are intended to provide a minimum set of operations into which the activities of a computation can be broken. These clearly do not include the extraction of a square root, which in most computers must be *programmed*, *i.e.*, built up from more elementary operations under the control of a sequence of instructions. Some quite powerful computing instruments have similarly dispensed with a wired-in division procedure. Carrying this spirit of parsimony a step further, the programmer may also be required to build up multiplication from the simpler arithmetic operations, which would thus seem to be only addition and subtraction. (Operations of a less sharply arithmetic character, like the familiar *extract*, can be accomplished if means for doubling and branching on the most significant bit are provided.) However, the list can be reduced still further by the omission of addition. By twice subtracting a number from itself, its negative may be formed. (In $M'AC$, as in many binary computers, the most significant bit of a word is regarded as having a negative position value.[2]) By use of this complemented number and the operation of subtraction, the effect of adding the original number may be accomplished.

The basic set of operations required for a gpc thus appears to be 1) subtract, 2) record in memory, and 3) branch. It is not necessary, however, that these operations have separate orders. In $M'AC$ each instruction execution accomplishes all three of these operations—a number read from memory (by head M) is subtracted from that held in the circulating register, the result is simultaneously recorded in memory (by head M^*), and the next instruction is then read either from M or from M^*, depending on the outcome of the subtraction. (Specifically, the criterion chosen is the state in which the *borrow flip flop*, B, is left following the number word of the execute phase.) Typically, one of these three operations will be the desired activity; the other two will be unneeded by-products. Thus, the simplification of reducing the order list to one item is obtained at the cost of less efficient use of memory capacity and greater complexity of the programmer's task.

Since only one order is provided, no bits of the instruction need be used for order specification. Accordingly, the entire instruction word is used for the delay number. The arithmetic unit, consisting of the borrow flip flop, B, and associated circuits, is used to effect the reduction of the delay number in instruction word periods of the delay phase, marked $\overline{N}\overline{E}$. This is accomplished by presetting B to the on state just prior to these word periods, at the times $N\overline{E}L$, and proceeding thereafter as if the number zero were being subtracted.

[2] S. P. Frankel, "The logical design," IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-6, pp. 5–14; March, 1957.

Three of the constituents of the complexity of the $M'AC$ design may now be summarized: except during the word periods marked NE, three bits of information, R, M, and M^*, are received from memory and one bit, R'', returned. In word periods NE two bits, R and M, are received and two returned, R'' and M''. The active state is determined by the two bits, E and B, together with the two bits of timing information, L and N.

Logical Design

In every word period the arithmetic unit is engaged in performing a subtraction, the result (remainder) of which is recorded as R'' except in word periods $\overline{N}E$. The remainder is also recorded as M'' in word periods NE. The minuend consists of the bits R at all times. The subtrahend bits are zero during the delay phase, M in word periods NE, and R in word periods $\overline{N}E$ (to ensure that no change in the state of B occurs then). Thus the subtrahend, denoted s, is expressed by $E(MN+R\overline{N})$. The borrow flip flop, B, is ordinarily turned on by a one-bit in the subtrahend unmatched in the minuend, off by a one in the minuend and zero subtrahend. Otherwise it remains unchanged, except that it is turned on at the end of a number delay word period (to provide for the diminution of the delay number) and off at the end of any instruction word period. Altogether, the condition for setting B on is $s\overline{R}+N\overline{E}L$, for turning it off is $\bar{s}R+\overline{N}L$. The remainder bits, denoted r, may be described as $R\oplus s\oplus B$ where $\oplus$ represents the *exclusive or* function.

The bits, R'', recorded in the circulating register are r except in word periods marked $\overline{N}E$, when they are M or M^* as B is on or off. They are thus described by the expression, $r(N+\overline{E})+\overline{N}E(BM+\overline{B}M^*)$. At the end of each instruction word period, hence on $\overline{N}L$, the bit R'' is set into flip flop E. The condition for recording in main memory is denoted w, and has the value $\overline{N}E$. The bits, M'', recorded then are the remainder bits, r.

This description of the logical design is summarized in Table I. The condition for turning on flip flop B is denoted B', that for turning it off is denoted $\overline{B}'$, etc.

TABLE I
M'AC Logic

Timing Signals	Logical Equations	Definitions
$N'=\overline{N}L$	$B'=s\overline{R}+N\overline{E}L$	$s\equiv E(MN+R\overline{N})$
$\overline{N}'=NL$	$\overline{B}'=\bar{s}R+\overline{N}L$	$r\equiv R\oplus s\oplus B$
	$R''=r(N+\overline{E})+\overline{N}E(BM+\overline{B}M^*)$	
	$E'=R''\overline{N}L$	
	$\overline{E}'=\overline{R}''\overline{N}L$	
	$w=\overline{N}E$	
	$M''=r$	

Iterative Combinational Switching Networks—General Design Considerations*

E. J. McCLUSKEY, Jr.†

Summary—**An iterative network is a combinational switching circuit which consists of a series of identical "cells" or sub-networks; for example, the stages of a parallel binary adder. A formal design method for iterative networks is presented. This is similar to the flow table technique for designing sequential circuits.**

1. Introduction

An iterative network is a combinational switching circuit which consists of a series of identical "cells" or sub-networks. Some typical iterative networks are shown in Fig. 1 and Fig. 2, and the general iterative network structure is shown in Fig. 3. There are two situations in which it is appropriate to design a circuit as an iterative network. The first is when the conditions for an output can be conveniently stated in terms of the number of inputs which are active,[1–4] for example, when there is to be an output only when 2 or 5 of the inputs are active. The second situation arises when the conditions for an output can be stated in terms of the relative positions of the inputs which are active. Examples of the second situation are: 1) an output is desired if any two adjacent inputs are active and

* Manuscript received by the PGEC, June 10, 1958; revised manuscript received, July 30, 1958.

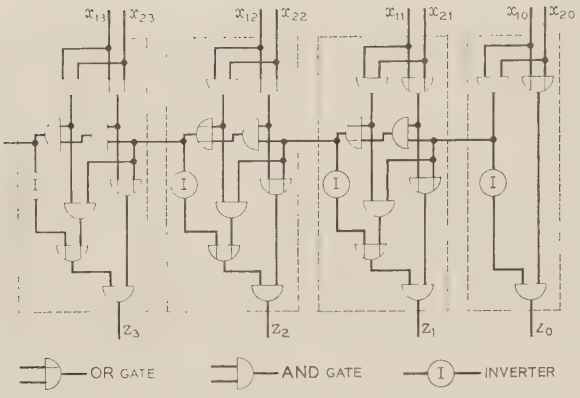
† Bell Telephone Labs., Whippany, N. J.; formerly with Princeton University, Princeton, N. J.

[1] In this case the desired switching function is said to be totally symmetric.

[2] W. Keister, A. E. Ritchie and S. H. Washburn, "The Design of Switching Circuits," D. Van Nostrand Co., New York, N. Y., pp. 55–64; 1951.

[3] S. H. Caldwell, "Switching Circuits and Logical Design," John Wiley and Sons, Inc., New York, N. Y., ch. 11; 1958.

[4] S. H. Washburn, "Relay 'trees' and symmetric circuits," Trans. AIEE, pt. I. vol. 68, pp. 582–586; 1949.



$$x_1 = 2^0\, x_{10} + 2^1\, x_{11} + 2^2\, x_{12} + 2^3\, x_{13}$$
$$x_2 = 2^0\, x_{20} + 2^1\, x_{21} + 2^2\, x_{22} + 2^3\, x_{23}$$
$$z = x_1 + x_2$$

Fig. 1—An iterative network for adding two binary numbers, x_1 and x_2.

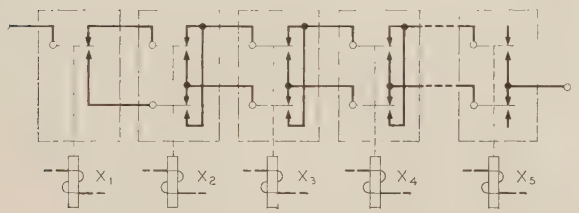


Fig. 2—A relay iterative network which is closed when an odd number of relays are operated.

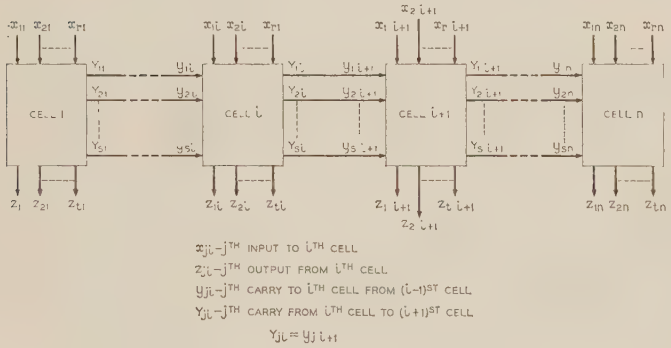


Fig. 3—General structure of an iterative network.

2) an output is desired if $X_1 = 1$ and $X_2 = 0$ for any cell i, *and* there has been no preceding cell j for which $X_1 = 0$ and $X_2 = 1$. A basic characteristic of the second situation is that some ordering among the inputs is assumed.

An iterative network usually requires less equipment and is easier to wire-up than a standard combinational circuit having the same switching function. This economy in equipment usually carries with it the penalty of a longer operating time.[5] Because use is made of the symmetry of a switching function, it is easier to design an iterative network than to carry out a general combinational circuit design, and it is possible to design functions of large numbers of variables.

[5] For electronic networks.

There are two general types of iterative networks: 1) Those which have outputs from each cell (Fig. 1), which will be called *cell-output networks;* and 2) those which have outputs only from the last cell, (Fig. 2) which will be called *circuit-output networks*. In general, the same design techniques apply to both types of network. Specific differences will be discussed elsewhere.

The idea of an iterative contact network was first discussed by Keister, Ritchie, and Washburn[6] (in this book the term reiterative circuits is used). A suggestion for a formal design procedure using sequential circuit flow tables was published by Huffman,[7] and a detailed discussion of both approaches is given by Caldwell.[8] These presentations are concerned almost entirely with contact networks and lack generality since they place restrictions on the nature of the allowed combinations of carry signals. In this paper a general method for designing both relay and electronic iterative networks is presented. This method makes use of an analogy between iterative networks and clocked sequential switching circuits.

II. Comparison of Iterative and Noniterative Combinational Circuits

Before starting the discussion of iterative network design, a comparison of iterative and noniterative networks will be made. This will be done by considering the design of a parallel binary adder. Since such an adder is a combinational circuit, it can be specified by means of a table of combinations[9] such as Table I(a). For adding together two two-bit numbers such a table has sixteen rows; in general for adding two n-bit numbers, 2^{2n} rows are required. It is clear that it would be quite laborious just to write down the table of combinations for a circuit to add together two numbers of useful length. The algebraic expressions corresponding to Table I(a) are listed in Table I(b). Fig. 4 shows the two-stage diode circuit which corresponds directly to these expressions. This circuit contains an "and" gate with four inputs. In general, a circuit to add together two n-bit numbers would require gates with $2n$-inputs. Gates with large numbers of inputs are difficult to realize physically, and may require the addition of more stages of logic. In an iterative network such as that shown in Fig. 1, the size of the gates is independent of the size of the numbers being added. There is an advantage to the network of Fig. 4 in that the only associated delay is that of a signal passing through two gates (an "and" gate and an "or" gate) in series. In the network of Fig. 1 there can be a delay due to two gates in series *per cell* so that the high-order output bits may

[6] *Op. cit.*, pp. 55–64.
[7] D. A. Huffman, "The Synthesis of Iterative Switching Circuits," *Quart. Prog. Rep.*, Res. Lab. of Elec., Mass. Inst. of Tech., Cambridge, pp. 63–67; January, 1955.
[8] *Op. cit.*, ch. 11.
[9] W. Keister, A. E. Ritchie and S. H. Washburn, *op. cit.*, p. 89.

TABLE I

SPECIFICATIONS FOR A CIRCUIT TO ADD TWO 2-BIT BINARY NUMBERS $X_1 = X_{11}X_{10}$ AND $X_2 = X_{21}X_{20}$ AND PRODUCE AN OUTPUT $Z = Z_2Z_1Z_0$

(a)

X_2		X_1		Z		
X_{21}	X_{20}	X_{11}	X_{10}	Z_2	Z_1	Z_0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

(b)

$$Z_0 = X_{20}X_{10}' + X_{20}'X_{10}$$
$$Z_1 = X_{21}X_{11}'X_{10}' + X_{21}X_{20}'X_{11}' + X_{21}'X_{20}'X_{11} + X_{21}'X_{11}X_{10}' + X_{21}'X_{20}X_{11}'X_{10} + X_{21}X_{20}X_{11}X_{10}$$
$$Z_2 = X_{21}X_{11} + X_{21}X_{20}X_{10} + X_{20}X_{11}X_{10}$$

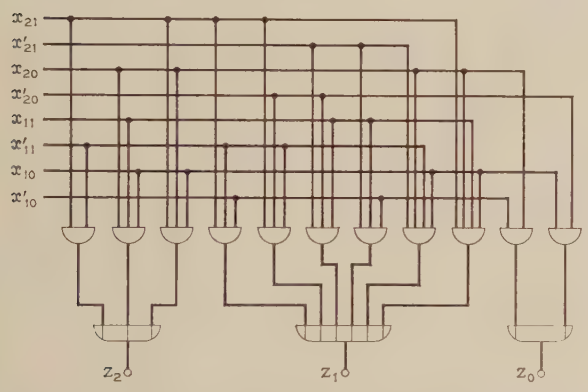
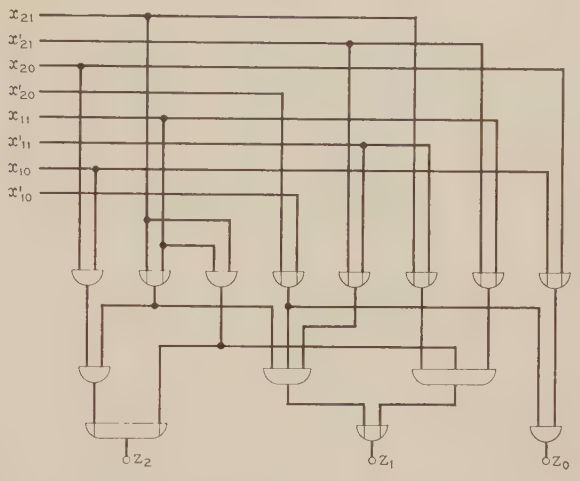


Fig. 4—Circuit to produce the sum of two two-bit binary numbers.

lag the inputs by an appreciable time. In summary, a two-stage electronic network is generally faster than an iterative circuit, but may require an excessive amount of equipment. The two-stage circuit is also quite laborious to design, although this design may be mechanized on a computer.[10]

By factoring the algebraic expressions for a two-stage logic network, it is possible to obtain other circuits which may require fewer diodes and which will have more of a delay through the circuit. One such factorization for the binary adder is shown in Fig. 5. In general, for any table of combinations there will be one factorization which will correspond exactly to the iterative circuit realization. The difficulties with designing a circuit by factorization are that no straightforward technique is known for determining just how to factor (this is still very much of an art, depending strongly on the skill of the designer)[11] and the expressions to be manipulated are extremely unwieldy. When applicable, the iterative design technique is useful for avoiding the difficulties mentioned in connection with the direct combinational circuit design method.



$$z_2 = x_{20}\,x_{10}\,(x_{21} + x_{11}) + x_{21}\,x_{11}$$
$$z_1 = (x'_{21} + x'_{11})(x_{21} + x_{11})(x'_{20} + x'_{10}) + x_{21}\,x_{11}\,(x_{21} + x'_{11})(x'_{21} + x_{11})$$
$$z_0 = (x_{20} + x_{10})(x'_{20} + x'_{10})$$

Fig. 5—Circuit to produce the sum of two two-bit binary numbers.

III. SYNTHESIS

Since in an iterative network all of the cells are identical except for the end cells, it is only necessary to specify enough information for the design of a typical cell and for the modifications in the end cells. The typical cells will be considered first. A (typical) cell must "determine" what signals to place on its output leads and on its carry leads to the next cell: The cell determines this on the basis of the signals which it receives on its input leads and on its carry leads from the preceding cell. The carry leads from the ith cell to the $i+1$st cell will be called the *i output-carries* and the carry leads from the i-1st cell to the ith cell will be called the *i input-carries*. A cell can be specified by a table which lists the signals on the output leads and the output-carry leads for each possible combination of signals on the input leads and the input-carry leads. Table II shows such a table for the binary adder of Fig. 1. The input lead combinations are listed across the top of the table and the input-carry lead combinations are listed along the side of the table. The corresponding output lead and output-carry lead conditions form the entries of the table. Such a table will be called an *output*

[10] E. J. McCluskey, Jr., "Minimization of Boolean functions," *Bell Sys. Tech. J.*, vol. 35, pp. 1417–1444; November, 1956.
R. H. Urbano and R. K. Mueller, "A topological method for the determination of the minimal forms of a Boolean function," IRE TRANS. ON ELECTRONIC COMPUTERS, vol. 5, pp. 126–132; September, 1956.

[11] T. J. Beatson, "Minimization of components in electronic switching circuits," TRANS. AIEE, vol. 37, pp. 283–291; July, 1958.

TABLE II
OUTPUT TABLE FOR A BINARY ADDER

x_1x_2 / y	0 0	0 1	1 1	1 0
0	0, 0	0, 1	1, 0	0, 1
1	0, 1	1, 0	1, 1	1, 0

Y, Z

$Y = y(x_2+x_1)+x_1x_2$
$Z = x_1'x_2'y+x_1'x_2y'+x_1x_2'y'+x_1x_2y$
$= (x_1+x_2+y)(Y'+x_1x_2y)$

table. The algebraic expressions for the outputs and output-carries can be determined directly from such a table by standard techniques.[12] This table is identical with the output table for a clocked (synchronous) sequential circuit in which the y-variables represent present-state variables and the Y-variables represent next-state variables.[13] Thus an analogy exists between iterative networks and clocked sequential circuits, and many of the techniques of sequential circuit design will be shown to apply to iterative networks.

An output table specifies the typical cell completely. For the first cell, an assumption must be made of what the input-carries should be. The proper assumption is clear from the requirements of the problem. In the case of the binary adder, the carry lead represents the arithmetic carry of ordinary addition: Clearly the carry to the first cell will be zero. This knowledge of what the carry must be can be used to simplify the design of the first cell. Similarly, in the last cell, the elements which generate the carry outputs can be eliminated since no carry outputs are necessary.

It has been shown that an iterative network is determined completely by an output table for which the input carries to the first cell are specified. The problem of designing an iterative network therefore can be reduced to that of discovering an output table which corresponds to a network having the desired properties. The following material discusses the formulation of output tables.

The first step in writing down an output table is to decide what "information" must be passed from cell to cell via the carry leads. In the binary adder the signal sent from the ith cell to the $i+1$st cell specifies whether an *arithmetic* carry has been generated in the ith cell. No formal procedure can be given for determining what information the carries must represent. Actually, in some sense no formal procedure is possible; writing the carry-output table corresponds to specifying precisely what is the desired behavior of the circuit. Similarly no "formal technique" can be given for writing the table of combinations for an ordinary combinational circuit. Some consolation can be derived from the fact that if more carry "information" is assumed than is actually necessary, the superfluous "information" is easily eliminated by a formal procedure, Section IV. Once the information which the carry leads must convey has been decided upon, it is a straightforward process to fill in the table. For each combination of input and input-carry lead signals, a decision must be made as to the correct output and output-carry signals. The entire procedure of forming an output table is best illustrated by an example. A circuit is to be designed which has inputs X_n, X_{n-1}, $\cdots$, X_1 and outputs Z_n, Z_{n-1}, $\cdots$, Z_1 such that $Z_i=1$ if and only if[14]

$$\sum_{j=1}^{i} X_j = 1 \text{ or } 2.$$

It must be possible to determine from the carry leads the following pieces of information:

0) None of the inputs (X_i) to preceding cells are equal to one.
1) One of the inputs (X_i) to a preceding cell is equal to one.
2) Two of the inputs to preceding cells are equal to one.
3) Three or more of the inputs to preceding cells are equal to one.

By analogy with sequential circuits each such possibility will be called a *carry state*.[12] Since the carry leads have binary signals, *at least* two leads are necessary to convey these four possibilities. At this stage of the design process it is unwise to assume a specific number of carry leads or to decide which signals on the carry leads will represent the different carry states. Instead, each carry state will be assigned a decimal number. A table will be formed which lists the input combinations across the top and the input-carry states along the side. The output-carry states and circuit outputs will be the entries of the table. (See Table III.) This table is called a *state table* or *flow table*. If the input-carry state is 1 (one previous $X_j=1$) and $X_i=0$, the next carry state is again 1 and $Z_i=1$ since there has been one X_j equal to 1. If the input-carry were 2 and $X_i=1$, the next carry would be 3 and $Z_i=0$ since then there would be three $X_j=1$. The *input-carry state will be represented by* s_k and the *output-carry state will be represented by* S_k where s_k and S_k represent decimal numbers.

The output table is derived directly from the state table by replacing each decimal number by an arbitrary binary number.[15] The binary numbers will correspond

[12] D. A. Huffman, "The synthesis of sequential switching circuits," *J. Franklin Inst.*, vol. 257, pp. 161–303; March, 1954.
[13] M. Phister, "Logical Design of Digital Computers," John Wiley and Sons, Inc., New York, N. Y.; 1958.
[14] The addition here is ordinary addition rather than Boolean addition.
[15] There are many possible choices for the binary numbers. Different choices will result in different networks. The problem of choosing the binary numbers with which to encode the states is discussed at the end of this section.

TABLE III

TABLES FOR AN ITERATIVE CIRCUIT FOR WHICH $Z_i=1$ IF, AND ONLY IF, $\sum_{i=1}^{i} X_i=1$ OR 2

(a) State Table

	s_k	$x_i=0$	$x_i=1$
0 preceding $x=1$	0	0, 0	1, 1
1 preceding $x=1$	1	1, 1	2, 1
2 preceding $x=1$	2	2, 1	3, 0
3 or more preceding $x=1$	3	3, 0	3, 0
		S_k, Z_i	

(b) Output Table

y_1y_2 \ X	0	1
01	0 1, 0	1 1, 1
11	1 1, 1	1 0, 1
10	1 0, 1	0 0, 0
00	0 0, 0	0 0, 0
	Y_1Y_2, Z_i	

$Y_1=x'y_1+xy_2$
$Y_2=y_2(x'+y_1')$
$Z_i=Y_1$

to the actual signals on the carry leads. Each occurrence of the same decimal number must be replaced by the same binary number. The output table of Table III(b) results from the state table of Table III(a) when the replacement of decimal by binary numbers is made according to the following table:

S_k	Y_1Y_2
s_k	y_1y_2
0	0 1
1	1 1
2	1 0
3	0 0.

The following algebraic expressions result from the output table of Table III(b): $Y_1=x_i'y_1+x_iy_2$, $Y_2=y_2(x_i'+y_1')$, $Z_i=Y_1$. The corresponding circuit is shown in Fig. 6. The typical cell is designed, using standard design techniques,[16] as a combinational diode logic circuit having inputs x_i, x_i', y_1, y_2 and outputs Y_1, Y_2, and Z_i. It is usually possible to simplify some of the end cells (cells 1, 2, $\cdots$ and n, $n-1$, $\cdots$) so that they require less equipment than typical cells. For the first cell, the input state is known. In the circuit whose design is being described, this first input state is 0: None of the preceding cells have $x_i=1$, since there are no preceding cells. State 0 was coded as $y_1=0$, $y_2=1$ so that these values can be substituted in place of y_1 and y_2 in the equations for Y_1, Y_2 and Z_1: $Y_{11}=x_1'\cdot 0+x_1\cdot 1=x_1$, $Y_{21}=1(x_1'+1)=1$, $Z_1=Y_{11}=x_1$. These equations lead to cell 1 of Fig. 6. For cell 2, $y_{12}=Y_{11}$, $y_{22}=Y_{21}=1$, so

[16] Caldwell, *op. cit.*, ch. 9.

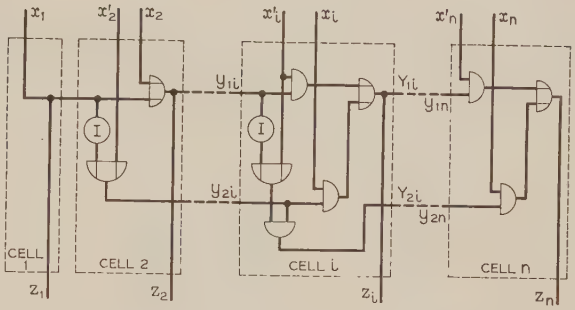


Fig. 6—Network for the output table of Table III.

that $Y_{12}=x_2'Y_{11}+x_2\cdot 1=Y_{11}+x_2$, $Y_{22}=1(x_2'+Y_{11}')=x_2'+Y_{11}'$. Cell 3 will be a typical cell since $y_{13}=Y_{12}$, $y_{23}=Y_{22}$. For the final cell (n) it is only necessary to generate Z_n since there is no next cell to which Y_{1n} and Y_{2n} must be connected. However, $Z_n=Y_{1n}$ requires both y_1 and y_2 so that cell ($n-1$) must generate both Y_{1n-1} and Y_{2n-1}.

This completes the design of the network, and illustrates the general design procedure. One problem connected with the design procedure which was not discussed is the *assignment problem*—the assignment of binary numbers to the states. For gate networks, any choice of the binary numbers will lead to a realizable network. This is not generally true for relay networks. It is always possible to obtain a physically realizable relay network by assigning to the states only binary numbers which contain only one 1. For example, a coding for the table of Table III(a) which would lead to a realizable relay network would be:

S_K	$Y_1Y_2Y_3Y_4$
s_k	$y_1\ y_2\ y_3\ y_4$
0	1 0 0 0
1	0 1 0 0
2	0 0 1 0
3	0 0 0 1

The resulting output table and carry functions are shown in Table IV and the network is shown in Fig. 7. For this table it turns out that y_4 is not necessary since neither Z nor Y_1, Y_2 or Y_3 depend on it.

A more complete discussion of the assignment problem will be presented elsewhere.

Just as it is possible to design sequential circuits from a state diagram[17,18] rather than a state table, it is also possible to use a state diagram in carrying out an iterative network design. The state diagram corresponding

[17] G. H. Mealy, "A method for synthesizing sequential circuits," *Bell Sys. Tech. J.*, vol. 34, pp. 1045–1080; September, 1955.

[18] E. F. Moore, "Gedanken-Experiments on Sequential Machines," *Automata Studies*, Annals of Mathematics Studies, no. 34, Princeton University Press, Princeton, N. J.; pp. 129–153; 1956.

TABLE IV
DERIVATION OF CARRY FUNCTIONS FOR RELAY REALIZATION OF STATE TABLE OF TABLE III(a)

(a) Output Table

$y_1y_2y_3y_4$	X 0	1
1000	1000, 0	0100, 1
0100	0100, 1	0010, 1
0010	0010, 1	0001, 0
0001	0001, 0	0001, 0
	$Y_1Y_2Y_3Y_4$, Z	

(b) Function

$Y_1=x'y_1$, $Y_2=x'y_2+xy_1$
$Y_3=x'y_3+xy_2$, $Z=Y_2+Y_3$

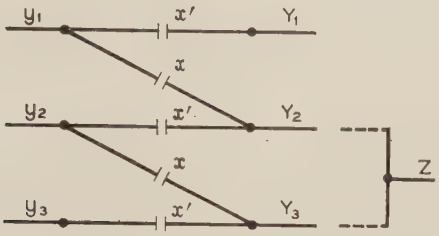


Fig. 7—A typical cell for the network for Table III. The end-cell connections are shown by dotted lines.

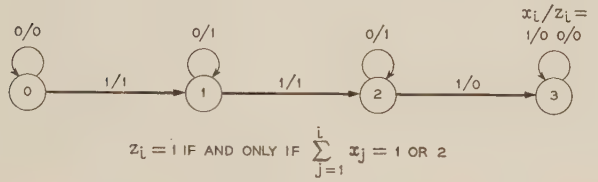


Fig. 8—State diagram for an iterative circuit.

to the state table of Table III is shown in Fig. 8. Only the state table approach will be discussed here.

IV. Reduction of the Number of States

It is possible in forming the state table to assume more carry-lead information than is actually required. This leads to superfluous states in the state table. The purpose of this section is to discuss formal procedures for eliminating such unnecessary states.

One reason that it is desirable to reduce the number of states to a minimum is that the reduced state table is less cumbersome and easier to manipulate. An even more important reason for reduction is that the required number of carry leads is roughly proportional to the number of states. Number of carry leads $\geq \log_2$ (number of states). In an electronic network it is usually necessary to provide an amplifier per cell for each carry lead. Reducing the number of carry leads thus reduces the number of amplifiers required. For both relay and electronic networks a reduction in the number of carry leads reduces the number of outputs which must be generated by the combinational circuitry in each cell. This may lead to a reduction in the combinational circuitry required; but such a reduction is not guaranteed since reducing the number of carries may at the same time complicate the combinational circuitry required for each output carry.

In summary, minimizing the number of states reduces the number of carry lead amplifiers required in an electronic network and may simplify the combinational circuitry in electronic and relay networks.

One type of superfluous state is a state which can never actually occur. Such a state is called an *inaccessible state*. It is possible to have, in a state table, an inaccessible state which occurs as an input-carry state, but which *does not* appear as an output-carry state and is not an initial state. Since the circuit does not start in the inaccessible state and does not go to the inaccessible state from any other state, the inaccessible state never actually occurs and can be removed from the state table specification of the circuit. It is also possible to have an inaccessible state which can be reached from other states, but cannot be reached starting from the specified initial state. For example, in the following flow table, states 3 and 4 are inaccessible if the specified initial state is 1.

	P_j	X 0	1
INITIAL STATE	1	1	2
	2	2	1
	3	3	4
	4	4	3.

It is also possible to reduce the number of states by merging groups of equivalent states.[12,17,18] Two states of a circuit A and B are said to be equivalent if it is possible to replace A by B, eliminating A, without changing the performance of the circuit. The following experimental procedure can be carried out to determine whether two states A and B are equivalent. Two copies of the circuit being considered are constructed. The input carry leads to a typical cell, i, are disconnected from the preceding cell, and signals corresponding to state A are placed in the input carry leads to cell i of one circuit, and signals representing state B are placed on the corresponding leads of the other circuit. All possible inputs are applied to the two circuits and the outputs are compared. If the two circuits always have identical outputs when they have identical inputs, states A and B are equivalent; otherwise they are not. The outcome of such an experiment may depend on the number of cells following typical cell i. For two states which are "unconditionally" equivalent, the outputs from the two circuits used in the experiment will be identical for any number of cells following cell i. Two states which produce identical outputs for only a restricted number of cells following cell i are said to be equivalent for an experiment of length l, where l equals the number of cells having identical outputs. Two states which are identical for an experiment of length l can be

combined only for cells which are within l cells of the final cell of the circuit. Combining such cells is one method of determining the end cell modifications for the final cells of the circuit.

Of course, it is not necessary to actually carry out such experiments in order to determine which states are equivalent. The outcome of such an experiment can be determined by inspection of the state table. For example, in the table below, states 1 and 2 are equivalent since they have the same outputs and go to identical output states. States 3 and 4 are not equivalent since they have different outputs. The definition of equivalent states given here is identical to that used for sequential circuits. Equivalent states will not be treated further since detailed discussions have been given elsewhere [2,17–18].

P_j	X 0	1
1	1, 0	4, 1
2	1, 0	4, 1
3	1, 1	4, 0
4	2, 0	3, 1.

A general method for designing iterative networks has been presented. Subsequent papers will present techniques for obtaining economical networks using various criteria of economy.

Some Properties of Boolean Equations*

N. ROUCHE†

Summary—**Solubility conditions for a set of Boolean equations are established, first with respect to one variable, then with respect to all variables. By consideration of relations between minimal terms, a simple matrix form is deduced for Boolean equations. Using finite group theory and the properties of the characteristic equation of the matrix, a classification is introduced for Boolean mappings and their iterations, to which corresponds a classification of sequential machines.**

Introduction

THE problem of solving Boolean equations has already been dealt with in the literature, but it does not seem that results of great generality have been obtained. See, for instance, a recent paper by Zemanek[1] which makes extensive use of truth-tables. It appears, however, that there exists a general criterion for solubility, expressible in terms of a determinant of ordinary algebra. Furthermore, there exists a matrix representation of Boolean equations which is free from difficulties previously encountered concerning multiplication and iteration of matrices.[2] It will be shown how the theory developed here throws some light on the problem of the structure of sequential machines.

Solubility of a System of Boolean Equations with Respect to One Variable

Let

$$y_i = g_i(x_1, x_2, \cdots, x_n) \qquad (i = 1, 2, \cdots, k) \quad (1)$$

be a system of k Boolean equations defining the k Boolean variables $y_1, y_2, \cdots, y_k$ in terms of the n Boolean variables $x_1, x_2, \cdots, x_n$. It is well known that (1) may be written as follows:

$$y_i = x_1\phi_{i1} + x_1'\phi_{i2} \quad (2)$$

where $+$ indicates union, the absence of symbol indicates intersection, and x_1' is x_1 complemented. Both ϕ_{i1} and ϕ_{i2} are Boolean functions of $x_2, x_3, \cdots x_n$.

We now consider under what condition it will be possible to express x_1 as a function of the y_i.

$$x_1 = f(y_1, y_2, \cdots, y_k). \quad (3)$$

The expression in the right member, reduced to canonical form, yields

$$f = \sum_j m_j(y_1, y_2, \cdots, y_k), \quad (4)$$

the right member being the sum of a certain number of minimal terms of the set of variables $y_1, y_2, \cdots, y_k$. But making use of (2), we have

$$m_j(y_1, y_2, \cdots, y_k) = x_1 m_j(\phi_{11}, \phi_{21}, \cdots, \phi_{k1}) + x_1' m_j(\phi_{12}, \phi_{22}, \cdots, \phi_{k2}).$$

Substituting into (4), we obtain

$$f = x_1 \sum m_j(\phi_{11}, \phi_{21}, \cdots, \phi_{k1}) + x_1' \sum m_j(\phi_{12}, \phi_{22}, \cdots, \phi_{k2}), \quad (5)$$

$\sum$ being in (4) and in (5) a summation on minimal terms of the same ranks. Therefore, we may rewrite (5)

$$f = x_1 f(\phi_{11}, \phi_{21}, \cdots, \phi_{k1}) + x_1' f(\phi_{12}, \phi_{22}, \cdots, \phi_{k2}). \quad (6)$$

Considering (6), we deduce the following theorem:

* Manuscript received by the PGEC, June 10, 1958.
† Lovanium University, Leopoldville, Belgian Congo, Africa.
[1] H. Zemanek, "Die Lösung von Gleichungen in der Schaltalgebra," *Arch. elect. Übertragung*, vol. 12, pp. 35–44; January, 1958.
[2] J. O. Campeau, "The synthesis and analysis of digitial systems by Boolean matrices," IRE Trans. on Electronic Computers, vol. EC-6, pp. 231–241; December, 1957.

Theorem 1: In order that x_1 be a given function $f(y_1, y_2, \cdots, y_k)$ it is necessary and sufficient that

$$f(\phi_{11}, \phi_{21}, \cdots, \phi_{k1}) = 1 \quad f(\phi_{12}, \phi_{22}, \cdots, \phi_{k2}) = 0. \quad (7)$$

The system (1) may also be written in the following form:

$$y_i = x_1\left(\sum_{j=1}^{2^{n-1}} m_j(x_2, \cdots, x_n)b_{ij}\right) + x_1'\left(\sum_{j=2^{n-1}+1}^{2^n} m_j(x_2, \cdots, x_n)b_{ij}\right) \quad (i = 1, 2, \cdots, k) \quad (8)$$

where each coefficient b_{ij} is 0 or 1, and $m_{j+2^{n-1}}(x_2, \cdots, x_n) = m_j(x_2, \cdots, x_n)$ for all $j \leq 2^{n-1}$. The minimal term of rank l of the set $(y_1, y_2, \cdots, y_k)$ may be written, according to (8),

$$m_l(y_1, y_2, \cdots, y_k) = x_1\left[\sum_{j=1}^{2^{n-1}} m_j(x_2, \cdots, x_n)m_l(b_{1j}, \cdots, b_{kj})\right] + x_1'\left[\sum_{j=2^{n-1}+1}^{2^n} m_j(x_2, \cdots, x_n)m_l(b_{1j}, \cdots, b_{kj})\right]. \quad (9)$$

Any function of $y_1, y_2, \cdots, y_k$, being a sum of some minimal terms of the form (9), admits of the following development:

$$f(y_1, y_2, \cdots, y_k) = x_1\left[\sum_{j=1}^{2^{n-1}} m_j(x_2, \cdots, x_n)f(b_{1j}, \cdots, b_{kj})\right] + x_1'\left[\sum_{j=2^{n-1}+1}^{2^n} m_j(x_2, \cdots, x_n)f(b_{1j}, \cdots, b_{kj})\right]. \quad (10)$$

The necessary and sufficient conditions for

$$x_1 = f(y_1, y_2, \cdots, y_k)$$

are obviously

$$\left.\begin{aligned}\sum_{j=1}^{2^{n-1}} m_j(x_2, \cdots, x_n)f(b_{1j}, \cdots, b_{kj}) &= 1\\ \sum_{j=2^{n-1}+1}^{2^n} m_j(x_2, \cdots, x_n)f(b_{1j}, \cdots, b_{kj}) &= 0\end{aligned}\right\} \quad (11)$$

and these two relations must be true, whatever the values given to $x_2, \cdots, x_n$. Therefore we may state the following theorem:

Theorem 2: In order that x_1 be a given function $f(y_1, y_2, \cdots, y_k)$, it is necessary and sufficient that

$$\left.\begin{aligned}f(b_{1j}, \cdots, b_{kj}) &= 1 \quad 1 \leq j \leq 2^{n-1}\\ f(b_{1j}, \cdots, b_{kj}) &= 0 \quad 2^{n-1} < j \leq 2^n\end{aligned}\right\}. \quad (12)$$

This set of conditions, compared with those of Theorem 1, has the advantage that it depends only on the constant coefficients b_{ij} and not on the quantities x_i.

It is noteworthy that in the present problem the minimal terms of the sets of variables

$$b_{1j}, b_{2j}, \cdots, b_{kj} \qquad 1 \leq j \leq 2^n$$

play an important role. We shall see shortly that it is rewarding to dispose these minimal terms in an ordered rectangular array according to the following rules: all the minimal terms of the set of subscript j are written under each other in a column. These columns are juxtaposed, from left to right, in order of increasing j. In each column the minimal terms are arranged downwards in the order suggested by the increasing sequence of the first 2^k binary numbers. For instance, the minimal terms of three variables a, b, c, would be written downwards in the following order:

$$a'b'c',\ a'b'c,\ a'bc',\ a'bc,\ ab'c',\ ab'c,\ abc',\ abc.$$

The rectangular array is characterized by the fact that it contains one 1 per column, all other terms being 0. We shall call it the *matrix* of the Boolean mapping (1). This term will be justified later. We represent the matrix by

$$B = [\beta l_j].$$

In the case $k = n = 2$, the equations of the mapping are

$$\left.\begin{aligned}y_1 &= b_{11}x_1'x_2' + b_{12}x_1'x_2 + b_{13}x_1x_2' + b_{14}x_1x_2\\ y_2 &= b_{21}x_1'x_2' + b_{22}x_1'x_2 + b_{23}x_1x_2' + b_{24}x_1x_2\end{aligned}\right\} \quad (13)$$

and

$$[\beta_{lj}] = \begin{pmatrix} b_{11}'b_{21}' & b_{12}'b_{22}' & b_{13}'b_{23}' & b_{14}'b_{24}' \\ b_{11}'b_{21} & b_{12}'b_{22} & b_{13}'b_{23} & b_{14}'b_{24} \\ b_{11}b_{21}' & b_{12}b_{22}' & b_{13}b_{23}' & b_{14}b_{24}' \\ b_{11}b_{21} & b_{12}b_{22} & b_{13}b_{23} & b_{14}b_{24} \end{pmatrix}. \quad (14)$$

The Equivalent Mapping on Minimal Terms

Momentarily, in order to prove the following theorem, we make use of a matrix similar to the A matrix introduced by J. O. Campeau[2]

$$A = [a_{lj}] \qquad \begin{matrix} l = 1, 2, \cdots, k \\ j = 1, 2, \cdots, 2^k \end{matrix}$$

which has k rows and 2^k columns. The elements of column j are uniquely defined by

$$j - 1 = a_{1j}2^{k-1} + a_{2j}2^{k-2} + \cdots + a_{kj}2^0. \quad (15)$$

In the case $k = 2$, we have

$$A = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}.$$

Now, the b_{ij}'s of (8) are given by

$$b_{ij} = \sum_l a_{il}\beta_{lj}$$

and therefore

$$y_i = \sum_j \sum_l a_{il}\beta_{lj}\xi_j$$

if ξ_j is the minimal term of rank j of the set $(x_1, \cdots,$

x_n). In the same way, we obtain

$$y_i' = \sum_j \sum_l a_{il}'\beta_{lj}\xi_j.$$

A typical minimal term of the set $(y_1, \cdots, y_k)$ is

$$\begin{aligned} &y_1y_2' \cdots y_gy_{g+1}' \cdots y_k \\ &= \left(\sum_j \sum_{l_1} a_{1l_1}\beta_{l_1j}\xi_j\right)\left(\sum_j \sum_{l_2} a_{2l_2}'\beta_{l_2j}\xi_j\right) \cdots \\ &\left(\sum_j \sum_{l_g} a_{gl_g}\beta_{l_gj}\xi_j\right)\left(\sum_j \sum_{l_{g+1}} a_{g+1,l_{g+1}}'\beta_{l_{g+1},j}\xi_j\right) \cdots \\ &\left(\sum_j \sum_{l_k} a_{kl_k}\beta_{l_k,j}\xi_j\right). \end{aligned}$$

This may also be written

$$\begin{aligned} &y_1y_2' \cdots y_gy_{g+1}' \cdots y_k \\ &= \sum_{l_1, l_2, \cdots, l_g, l_{g+1}, \cdots, l_k} a_{1l_1}a'_{2l_2} \cdots a_{gl_g}a_{g+1,l_{g+1}}' \cdots a_{kl_k} \\ &\left(\sum_j \beta_{l_1j}\xi_j\right)\left(\sum_j \beta_{l_2j}\xi_j\right) \cdots \\ &\left(\sum_j \beta_{l_gj}\xi_j\right)\left(\sum_j \beta_{l_{g+1}j}\xi_j\right) \cdots \left(\sum_j \beta_{l_kj}\xi_j\right). \end{aligned} \tag{16}$$

Among all products of a_{il} in this expression, the only one which is different from 0 (and equal to 1) corresponds to

$$l_1 = l_2 = \cdots = l_g = l_{g+1} = \cdots = l_k = l$$

where l is the rank of the minimal term which is computed. Therefore (16) may be reduced to the very simple form

$$\zeta_l = \sum_j \beta_{lj}\xi_j. \tag{17}$$

Using matrix and vector notations, we write the formula

$$\zeta = B\xi. \tag{18}$$

This relation shows why we have called B a matrix. Using the rules of Boolean algebra, we have shown that (18) can be deduced from (1) or (8). Conversely, it is a very simple matter to show that (1) or (8) can be deduced from (18). We have

$$(y_1, y_2, \cdots, y_k) = A\zeta$$

or, using (18) we have

$$(y_1, y_2, \cdots, y_k) = AB\xi. \tag{19}$$

It is readily seen that

$$AB = \begin{bmatrix} b_{11} & b_{12} \cdots b_{1,2^n} \\ \cdots\cdots\cdots \\ b_{k1} & b_{k2} \cdots b_{k,2^n} \end{bmatrix} \tag{20}$$

and therefore (19) is equivalent to (1).

Theorem 3: To each mapping of n Boolean variables $x_1, \cdots, x_n$, into k variables $y_1, \cdots, y_k$ corresponds an equivalent mapping of the 2^n minimal terms of the x_j's into the 2^k minimal terms of the variables y_j. This equivalent mapping may be represented in matrix form, the terms of the matrix being simple combinations of the coefficients in the equations of the original mapping. Any Boolean mapping may be studied on the equivalent mapping on minimal terms. It is noteworthy that (18) *may be interpreted in ordinary algebra as well as in Boolean algebra.* This is due to the fact that the vectors of minimal terms never have more than one component equal to 1, all others being 0. Therefore, the only typical Boolean relation $1+1=1$ never appears in (18). We shall make use of this fact later.

Bi-Uniform Mappings of n Variables into n Variables

In this case, the matrix B is a square matrix. For the mapping to be bi-uniform, it is necessary and sufficient that the correspondence between minimal terms of the two sets of variables be a one-one correspondence. We therefore obtain the following theorem.

Theorem 4: A system of Boolean equations like (1) with $k=n$ may be solved for $x_1, x_2, \cdots, x_n$ if and only if the corresponding matrix B is nonsingular in the sense of ordinary matrix theory.

In this case, B contains one 1 per row and per column. It is a permutation matrix, and Theorem 5 results.

Theorem 5: Any bi-uniform mapping of n variables $x_1, x_2, \cdots, x_n$, into n variables $y_1, y_2, \cdots, y_n$ is equivalent to the equality of the vector of minimal terms in $y_1, y_2, \cdots, y_n$ and the vector obtained by a certain permutation of the components of the vector of minimal terms in $x_1, x_2, \cdots, x_n$.

As the vector of minimal terms has 2^n components, we may also state the following:

Theorem 6: There exist exactly $(2^n)!$ bi-uniform mappings of n variables into n variables.

An interesting necessary condition of bi-uniformity is suggested by the form of B. For a system like (1), with $k=n$ to be solvable with respect to $x_1, x_2, \cdots, x_n$, it is necessary that all the right members of the equations consist of 2^{n-1} minimal terms.

Obviously the set of all matrices corresponding to bi-uniform mappings of n variables into n variables forms a group of order $(2^n)!$, isomorphic with the symmetric group of degree 2^n. We shall call this group G.

Subgroups of the Group G of Bi-Uniform Mappings

At this point, it will be interesting to survey some of the subgroups of G to obtain a classification of all possible bi-uniform mappings.

Let us call centro-symmetric every permutation matrix which is symmetric with respect to its center. In order to deal more easily with centro-symmetric matrices, we momentarily introduce a new convention concerning subscripts of rows and columns for permutation matrices. If we are concerned with a matrix of even

order n^2, the subscripts i for the rows will be all the odd multiples of 0.5 taken in increasing order and satisfying

$$-\frac{n-1}{2} \leq i \leq \frac{n-1}{2}. \tag{21}$$

A similar convention is adopted for the column subscripts. The following is an example of a centro-symmetric permutation matrix of order 8^2 with the subscripts explicitly indicated.

	−3.5	−2.5	−1.5	−0.5	0.5	1.5	2.5	3.5
−3.5	1	0	0	0	0	0	0	0
−2.5	0	0	0	0	1	0	0	0
−1.5	0	1	0	0	0	0	0	0
−0.5	0	0	0	0	0	1	0	0
0.5	0	0	1	0	0	0	0	0
1.5	0	0	0	0	0	0	1	0
2.5	0	0	0	1	0	0	0	0
3.5	0	0	0	0	0	0	0	1

In the case of a matrix of odd order n^2, the subscripts will be all the integers, including zero, belonging to the closed interval determined by (21).

The condition for a matrix $[a_{ij}]$ to be centro-symmetric is then

$$a_{ij} = a_{-i,-j}. \tag{22}$$

The condition for symmetry (in the usual sense, *i.e.*, with respect to principal diagonal) is

$$a_{ij} = a_{ji}. \tag{23}$$

The condition for secondary symmetry (*i.e.*, symmetry with respect to the other diagonal) is

$$a_{ij} = a_{-j,-i}. \tag{24}$$

Eqs. [(22)–(24)] show that any symmetric centro-symmetric matrix displays secondary symmetry.

Theorem 7: The set of centro-symmetric permutation matrices of order n^2 is a subgroup of the group of permutation matrices of the same order.

The only axiom to be verified is the axiom of closure. Suppose that A and B are centro-symmetric. Let us prove that $C=AB$ is centro-symmetric. We have

$$c_{kl} = \sum_{j=-(n-1)/2}^{(n-1)/2} a_{kj}b_{jl} \tag{25}$$

and

$$c_{-k,-l} = \sum_{j=-(n-1)/2}^{(n-1)/2} a_{-k,j}b_{j,-l}. \tag{26}$$

But since A and B are centro-symmetric, (25) yields

$$C_{kl} = \sum_{j=-(n-1)/2}^{(n-1)/2} a_{-k,-j}b_{-j,-l}. \tag{27}$$

The summation being commutative and the interval covered by the summation subscript being symmetric with respect to 0, we may change $-j$ into j in (27) which gives

$$C_{kl} = \sum_{j=-(n-1)/2}^{(n-1)/2} a_{-k,j}b_{j,-l}. \tag{28}$$

Comparing (26) and (28), we get $C_{-k,-l}=C_{kl}$. We shall call this subgroup G_1. It is readily verified on an example that G_1 is in general nonabelian.

Theorem 8: The order of G_1 is $N=2^n(n/2)!$ To construct all centro-symmetric matrices of order n^2, we first put a 1 in the first row. There are n possibilities. At that moment, because of the condition of centro-symmetry, the last row is already occupied. There remains $n-2$ possibilities for the second row, etc. Thus

$$N = n(n-2)(n-4)\cdots 4.2$$

then

$$\frac{N}{2^{n/2}} = \frac{n}{2}\left(\frac{n}{2}-1\right)\left(\frac{n}{2}-2\right)\cdots 2.1 = \left(\frac{n}{2}\right)!$$

and

$$N = 2^{n/2}\left(\frac{n}{2}\right)!. \tag{29}$$

In the study of Boolean mappings, we are interested in matrices of order 2^{2n}. For such matrices

$$N = 2^{2^n-1}(2^{n-1})!. \tag{30}$$

Let us consider again the group G of all permutation matrices of order 2^{2n} corresponding to the permutations of a vector of minimal terms with 2^n components. Reversing the complementation of certain variables in the minimal terms is equivalent to permuting them in a certain manner. Thus G contains a certain number of matrices whose application to a vector of minimal terms merely reverses the complementation of some variables. We shall call these matrices, *matrices of complementation reversals.*

Theorem 9: The matrices of complementation reversals form an abelian subgroup G_{11} of order 2^n of G_1. The order that we have adopted for minimal terms in a vector of minimal terms is such that the following property holds. Two minimal terms in a symmetric position with respect to the center of the vector may be obtained from each other by complementation of all variables. And, conversely, if we reverse the complementation of some variables in all the minimal terms, the same property obviously still holds. Thus, all matrices of complementation reversals are centro-symmetric.

The axiom of closure is satisfied because it is clear that the product of two complementation reversals is a complementation reversal. There are 2^n distinct ways of reversing some complementations; thus the order of the subgroup is 2^n. Reversals of complementations are commutative, *i.e.*, the subgroup is abelian. The product

of two identical reversals of complementation is the identical operation. Thus, every matrix of G_{11} is also symmetric, and therefore secondary symmetric. This also shows that G_{11} is generated by n subgroups of order 2.

One of the characteristics of the matrices of G_{11} is that, except for the identity matrix, the elements at both ends of the principal diagonal are always 0, the first and the last minimal terms of a vector changing in every complementation reversal.

Let us now consider another type of permutation matrix. A certain number of the matrices of G correspond to a permutation of the original Boolean variables. There are $n!$ such permutations, which are equivalent to changing the names of the variables, and do not alter the structural character of the mapping. The matrices of this type will be called *variables permutation matrices.*

Theorem 10: *The variables permutation matrices contained in G form a nonabelian subgroup G_{12} of order n! of G_1.* First, it is obvious that these matrices form a subgroup of G because the product of two permutations of the variables is a permutation of the variables. Furthermore, this subgroup is isomorphic with the symmetric group of degree n, and therefore is nonabelian.

G_{12} is a subgroup of G_1. All variables permutation matrices are centro-symmetric because two minimal terms in a symmetric situation with respect to the center of the vector are transformed by a variables permutation into two minimal terms having the same property.

One of the characteristics of the matrices of G_{12} is that the elements at both ends of the principal diagonal are always 1, since the first and the last minimal terms of a vector never change in a variables permutation. This property and the fact that G_{11} has the inverse property prove that G_{11} and G_{12} have but a single element in common.

Theorem 11: *The product $G_2 = G_{11}G_{12}$ of the complementation reversals group and the variables permutation group is a subgroup of order $2^n n!$ of the centro-symmetric group.* For all the subgroups so far identified in G, we compute Table I. This shows that for $n=1$ and $n=2$, G_2 is an improper subgroup of G_1, and for $n>2$, G_2 is a proper subgroup.

TABLE I

Number of variables	n	1	2	3	4
Order of G	$(2^n)!$	2	24	40, 320	20, 924, 789, 888, 000
Order of G_1	$2^{2^n-1}(2^{n-1})!$	2	8	384	10, 320, 920
Order of G_2	$2^n n!$	2	8	48	384
Order of G_{11}	2^n	2	4	8	16
Order of G_{12}	$n!$	1	2	6	24

It is possible to develop still further the theory of permutation matrices and of singular Boolean mappings square and rectangular matrices from an abstract point of view, but it will be more convenient for our purpose to go in on this direction after having introduced the notion of sequential machine which will constitute a suggestive example of application of Boolean mappings to a concrete case.

Autonomous Sequential Machines and Flow-Graphs

We consider a machine with n terminals, each of them being capable of two states (for instance low and high potentials). The machine receives a discrete sequence of advancing pulses, and it may change the states of its terminals at each advancing pulse. If none of the n terminals is used for arbitrary input signals introduced before the advancing pulses, we call such a machine an autonomous sequential machine. As has already been shown,[2] the y_i's of (1) with $k=n$ may be interpreted as giving the state of the machine after advancing pulse t, if the x_i's give the state of the machine before the same pulse. All the properties of the machine are then contained in the matrix B of the mapping. The properties of the matrix may be visualized in a flow-graph, each state of the machine being represented by a small circle containing the number of the state, that is, the rank of the corresponding minimal term. If, before an advancing pulse, the machine is in state V, its state after the advancing pulse will correspond to the row subscript of the single 1 of column V in the matrix.[3]

For instance, suppose a machine is defined by the following:

$$\left.\begin{aligned} y_1 &= x_1x_2'x_3 + x_2x_3 + x_2'x_3' \\ y_2 &= x_1'x_2'x_3' + x_1x_3' + x_1'x_3 \\ y_3 &= (x_1'x_2' + x_1x_2)x_3 \end{aligned}\right\}; \tag{31}$$

its matrix is

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{32}$$

and its flow-graph is shown in Fig. 1.

Reversible Autonomous Sequential Machine

We call reversible every machine whose matrix B is nonsingular. To every 1 situated on the principal diagonal, there corresponds a distinct part of the flow-

[3] It will be recognized here that the matrix B is similar to the transition matrix defined in the following reference. F. E. Hohn, S. Seshu, and D. D. Aufenkamp, "The theory of nets," IRE Trans. on Electronic Computers, vol. EC-6 ,pp. 154–161; September, 1957.

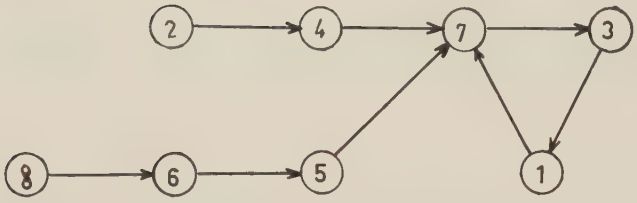


Fig. 1—Flow-graph of a machine.

Fig. 2—One—state circle.

graph consisting of a single state circle, as shown in Fig. 2. It is possible to find a permutation matrix P which will bring up this 1 into leading position on the principal diagonal if B is transformed by P. The matrix B is then transformed to

$$B_1 = PBP^{-1}. \tag{33}$$

If there is another 1 on the principal diagonal, a transformation of B_1 by a permutation matrix will bring this 1 into second position on the principal diagonal, and so on. When all the 1's of the principal diagonal are exhausted, suppose there is in the matrix a nonsingular principal minor of order 2. Transformation by an appropriate permutation matrix will bring this minor next to the line of 1's already treated. This minor corresponds to a double state circle in the flow-graph, as shown in Fig. 3.

More generally, we rearrange the matrix as a succession of nonsingular principal minors, as shown in Fig. 4, each principal minor containing no nonsingular minors. Each separated principal minor of order k in this representation has a corresponding k-state circle in the flow-graph.

To every such principal minor are associated k roots of the characteristic equation of the matrix, and these roots are given by

$$(-\lambda)^k - 1 = 0$$

if k is even, or by

$$(-\lambda)^k + 1 = 0$$

if k is odd; that is, in both cases, the k roots are

$$l^{\lambda(2\pi j/k)} \qquad \lambda = 1, 2, \cdots, k.$$

From this we deduce Theorem 12.

Theorem 12: The multiplicity of the root $l^{2\pi i/k}$ (where k is an integer) of the characteristic equation of B is equal to the number of circles of the flow graph whose degrees admit k as a divisor.

In particular, the multiplicity of the root 1 is equal to the number of distinct circles of the flow graph.

We shall call *simply-connected* any machine for which the multiplicity of the root 1 is equal to 1.

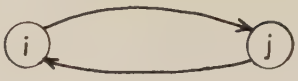


Fig. 3—Two—states circle.

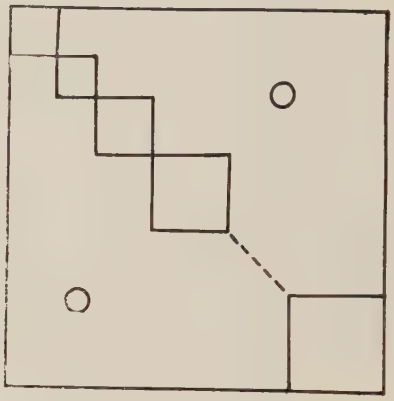


Fig. 4—Ordered permutation matrix.

Irreversible Autonomous Sequential Machines

Theorem 13: Every matrix B has at least one principal minor different from 0.

The matrix B is characterized by the fact that it has one and only one 1 per column. Suppose the determinant of B is 0. Then there is at least one row of zeros. Let us remove from the matrix this row and the column of same rank. We are left with a principal minor. If this minor is 0, it contains at least one row of zeros. We remove this row and the corresponding column, etc. If all principal minors found by this procedure are 0, we are at last left with a principal minor containing only one element, and this element is 1.

Let us now look for an ordered form for B in case it is singular. We choose one of the nonzero principal minors of least order, say A_0, and transforming B by a permutation matrix, bring this minor into leading position on the principal diagonal (Fig. 5). If there is no 1 in the rectangle A_1 we bring the next nonzero principal minor into second position on principal diagonal. If A_1 contains a certain number of 1's we bring them close to A_0 in A_2 using a permutation matrix (Fig. 6). If there is no 1 in A_3, we bring the next nonzero principal minor with its upper left corner at point x. Otherwise, using a permutation matrix, we bring all the 1's of A_3 in the left part of A_3 that we call A_4, etc.

At last, we are left with a matrix B, ordered as shown in Fig. 7, where each square on the principal diagonal is as shown in Fig. 8.

It is clear that ordering the matrix B has not altered its flow graph. Theorem 14, which follows, remains true for irreversible machines. The number of rows of zeros in B is equal to the number of transients that the machine may exhibit, *i.e.*, to the number of states that may be adopted as initial states in the operation of the machine, but will never be attained in course of opera-

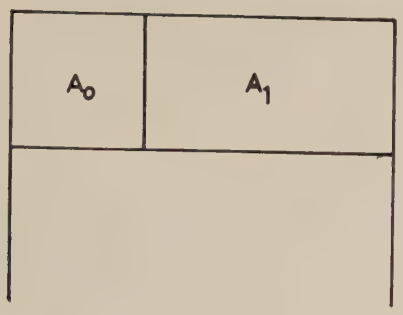


Fig. 5

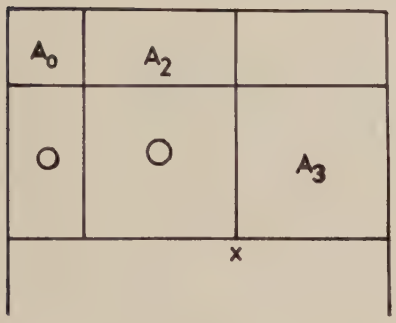


Fig. 6

tion. It is evident from Fig. 8 that the number of rows of zeros in B is equal to the multiplicity of the root 0 of the characteristic equation of B, and Theorem 14 results.

Theorem 14: The multiplicity of the root 0 *in the characteristic equations of B is equal to the number of possible distinct transients the machine may exhibit.*

Isomorphism of Machines

As has been shown, pre or post-multiplication of B by a matrix of the Group G_2 only changes the names or complementation of the variables but does not alter the structural character of the Boolean mapping. Thus, except for some possible inverters at inputs and outputs, the circuit may be the same for B and for B multiplied by a matrix of G_2. We shall call *isomorphic in a restricted sense* two machines whose matrices B_1 and B_2 may be obtained from each other by pre or post-multiplication by a matrix of G_2. Two such machines may be constructed with the same arrangement of elementary logical circuits.

On the other hand, transformation of B by a permutation matrix does not alter the character of the corresponding flow graph. We shall call *isomorphic in a broad sense* two machines whose matrices B_1 and B_2 may be obtained from each other through transformation by a permutation matrix. Two such machines will have the same flow graph, except for the names of the states. On the other hand, they may correspond to two altogether different Boolean mappings.

For instance, the ordered form of the matrix (32) is

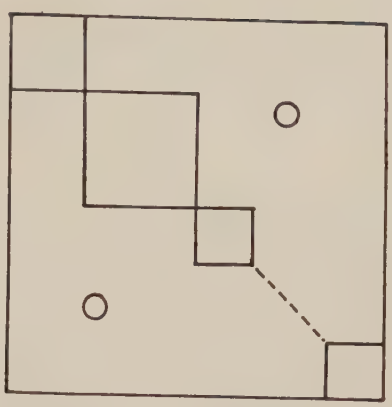


Fig. 7—Ordered singular matrix.

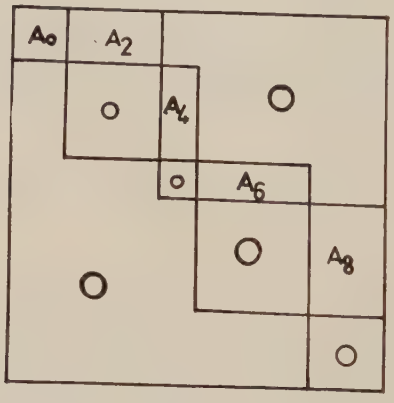


Fig. 8—Typical principal minor of ordered singular matrix.

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{34}$$

The corresponding equations are

$$\left.\begin{aligned} y_1 &= x_1x_3 \\ y_2 &= x_1'x_2'x_3 + x_1x_2x_3' \\ y_3 &= x_1x_2x_3' + x_2x_3 + x_2'x_3' \end{aligned}\right\} \tag{35}$$

and the flow graph is shown in Fig. 9.

Thus we may say that, to some extent,[4] the problem of minimizing a sequential machine whose flow graph

[4] The minimization of a sequential machine may be considered as consisting of three steps.
1) Minimization of the flow graph by reduction of equivalent states. (See D. D. Aufenkamp's and F. E. Hohn's, "Analysis of sequential machines," IRE Trans. on Electronic Computers, vol. EC-6, pp. 276–285; December, 1957.)
2) Determination among the possible sets of Boolean equations of the set which lends itself to the best minimization.
3) Actual minimization of the set of Boolean equations.

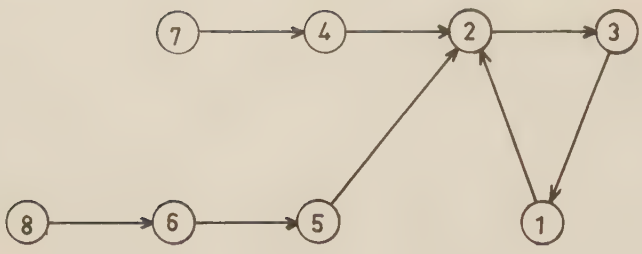


Fig. 9—Flow-graph of machine defined by matrix (34).

is given, is reduced to that of finding the permutation matrix P which, transforming B, will yield another matrix B corresponding to the simplest possible set of Boolean equations. We shall make no attempt here to solve this minimization problem.

Sequential Machines with External Input

We take as representing the equation of such a machine the following set of Boolean equations:

$$\left.\begin{aligned} y_1 &= f_1(x_1, \cdots, x_{n_1}, x_{n_1+1}, \cdots, x_{n_2}) \\ &\cdots\cdots\cdots\cdots\cdots\cdots \\ y_{n_1} &= f_{n_1}(x_1, \cdots, x_{n_1}, x_{n_1+1}, \cdots, x_{n_2}) \end{aligned}\right\} \tag{36}$$

where x_i $(i=1, \cdots, n)$ are the states of the output terminals before advancing pulse number n, and x_i $(i=n_1+1, \cdots, n_2)$ are the states of the input terminals, established before advancing pulse number n. The y_i's are the state of the output terminals after advancing pulse number n.

Eq. (36) may also be written

$$\left.\begin{aligned} y_1 &= \sum_{i=1}^{2^{n_1}} m_i(x_1, \cdots, x_{n_1}) L_{1i}(x_{n_1+1}, \cdots, x_{n_2}) \\ &\cdots\cdots\cdots\cdots\cdots\cdots \\ y_{n_1} &= \sum_{i=1}^{2^{n_1}} m_i(x_1, \cdots, x_{n_1}) L_{n_1 i}(x_{n_1+1} \cdots, x_{n_2}) \end{aligned}\right\}. \tag{37}$$

With (37) the machine appears similar to an autonomous sequential machine with the important difference that the flow-graph may be changed by exterior signals before each advancing pulse.

Eq. (36) may also be written

$$\left.\begin{aligned} y_i &= \sum_{i=1}^{2^{n_2}} m_i(x_{n_1+1}, \cdots, x_{n_2}) M_{1i}(x_1, \cdots, x_{n_1}) \\ y_{n_1} &= \sum_{i=1}^{2^{n_2}} m_i(x_{n_1+1}, \cdots, x_{n_2}) M_{n_1 i}(x_1, \cdots, x_{n_1}) \end{aligned}\right\}. \tag{38}$$

With (38), the machine appears to give a definite response to a given excitation except that it is capable of changing its response by taking the previous response into account. Both types of machines are defined by (36) and correspond to the same machine as seen from two different viewpoints.

Considering (37), we may define a matrix B as for the case of an autonomous machine. But now, B will occasionally change before each advancing pulse. The operation of an autonomous machine was given by the iteration of the matrix B. More precisely we had

$$\xi(n+k) = B^k \xi(n)$$

where ξ is the vector of minimal terms of the terminals, and the argument of ξ is the rank of the advancing pulse (*i.e.*, the time). In the present case, we have

$$\xi(n+k) = B(n+k-1) \cdots B(n)\xi(n)$$

and B has to be given as function of time.

Suppose this function is periodic of period k. Let

$$B_1 = B(n+k-1) \cdots B(n). \tag{39}$$

Then

$$\xi(n+jk) = B_1{}^j(n) \quad (j = 1, 2, \cdots).$$

Theorem 15: The operation of a sequential machine with a periodic input of period k, when it is observed at every kth advancing pulse, is the same as that of an autonomous machine with matrix $B=B_1$.

The complete operation of the machine is given by the matrices $B_1, B_2, \cdots, B_k$ corresponding to the k circular permutations of the terms in the second member of (39).

Conclusion

It has been shown how every set of Boolean equations corresponds to another set of equations in which the variables are the minimal terms of the variables in the initial system. This equivalent system of equations lends itself very easily to a matrix representation, the characteristic equation of the matrix giving important informations concerning the structure of the Boolean mapping considered and its iteration.

It seems likely that the methods developed in the present paper will lead to more results concerning classification, analysis, and synthesis of sequential machines.

Analysis of Sequential Machines II*

D. D. AUFENKAMP†

Summary—Mealy's model of a sequential machine is assumed and a relation of "compatibility" of states is introduced to further the analysis of such machines. In the event that input restrictions exist it is often possible to effect combinations of states under this relation in addition to those permitted under equivalence of states, a relation previously studied. Compatibility of states is analyzed by an iterative technique, rigorously established, which makes it possible to determine readily connection matrices of simpler "compatible" machines.

I. Introduction

In a previous paper,[1] we investigated the "equivalence" of states in Mealy's model of a sequential machine. That analysis yielded an iterative technique for finding all sets of equivalent states.

For machines in which there are input restrictions it is often possible to effect combinations of states in addition to those possible under equivalence without altering the relationships between inputs and outputs, as long as only allowable input sequences are presented to the machine. In this paper we investigate the relation of "compatibility" of states which permits this additional simplification of state diagrams. The notation and conventions established in the earlier paper[1] will also be observed in the following discussion. The problem discussed here has also been studied by Huffman.[2]

We recall the definition of Mealy's model of a sequential machine and certain other definitions which play an important role in the subsequent development.

Mealy's model of a sequential machine[3] assumes that the machine may be described in terms of:

1) a finite number of *states*, $s_1, s_2, \cdots, s_n$,
2) a finite number of distinct *inputs* symbolized by $x_1, x_2, \cdots, x_m$,
3) a finite number of distinct *outputs* symbolized by $y_1, y_2, \cdots, y_p$.

It is assumed that the *present output* and the *next state* are uniquely determined by the *present state* and the *present input*. Each such machine M may be represented by a net[4,5] called the *state diagram* of M in the following way: the set of vertices $v_1, v_2, \cdots, v_n$ of the net corresponds in 1-1 fashion to the set of states $s_1, s_2, \cdots, s_n$ of M. For each input which effects a transition of the machine from state s_i to state s_j, there is a direct branch from the vertex v_i to the vertex v_j. The ordered pair consisting of the input symbol concerned and the symbol for the corresponding output is assigned to this branch as its weight. When there is more than one such branch, say k branches from a vertex v_i to a vertex v_j, we simplify matters by employing only one branch and assign it a weight $(x_{r_1}, y_{r_1}) \vee (x_{r_2}, y_{r_2}) \vee \cdots \vee (x_{r_k}, y_{r_k})$ (where $\vee$ means "or") which defines the union of the weights of all the original branches from v_i to v_j.

The *connection matrix* of the machine is defined as follows:[6,7]

$$C = [c_{ij}]$$

where c_{ij} is the above defined union of the branch weights on all branches from vertex i to vertex j. If no branches connect vertex i to vertex j, then $c_{ij} = 0$.

By an *input sequence* we mean a sequence of input symbols, for example, $x_i, x_k, \cdots, x_j$. We say that a machine has *input restrictions*[8] if its output and next state are not defined for every input. Any input sequence which, if applied to the machine when it is in state i, violates no input restriction of state i or of any subsequent state, is called *an allowable input sequence for state i*.[9]

II. Fundamental Definitions

Definition 1

Two states, s_i of a machine M and t_j of a machine N, are called compatible states if and only if, starting with M in S_i and N in t_j, the two machines will yield identical output sequences on being presented with any given input sequence from the intersection of the set of allowable input sequences associated with state s_i and the set associated with state t_j. The output sequences associated with distinct input sequences need not, of course, be the same. An important special case occurs when the intersection is the null set, *i.e.*, when the two states have no common allowable input sequences. At the other extreme, if the two sets of allowable input sequences are identical, then compatibility of states becomes equivalence of states.[9]

* Manuscript received by the PGEC, June 20, 1958. This work has been carried out under the Lockheed General Res. Program.

† Missile Systems Div., Lockheed Aircraft Corp., Palo Alto, Calif.

[1] D. D. Aufenkamp and F. E. Hohn, "Analysis of sequential machines," IRE Trans. on Electronic Computers, vol. EC-6, pp. 276–285; December, 1957.

[2] D. A. Huffman, "The synthesis of sequential switching circuits," *J. Franklin Inst.*, vol. 257, pp. 161–190, March, 1954; pp. 275–303, April, 1954.

[3] G. H. Mealy, "A method for synthesizing sequential circuits," *Bell Sys. Tech. J.*, vol. 34, pp. 1045–1079; September, 1955.

[4] F. E. Hohn, S. Seshu, and D. D. Aufenkamp, "The theory of nets," IRE Trans. on Electronic Computers, vol. EC-6, pp. 154–161; September, 1957.

[5] Mealy, *op. cit.*, p. 1052.

[6] Hohn, Seshu, and Aufenkamp, *op. cit.*, pp. 156, 158.

[7] Aufenkamp and Hohn, *op. cit.*, p. 276.

[8] *Ibid.*, p. 277.

[9] *Ibid.*, p. 278.

Definition 2

Two machines M and N are called *compatible machines* if and only if for each state s_i of M there exists at least one compatible state t_j of N, and for each state t_j of N there exists at least one compatible state s_i of M.

Except in special cases, compatibility of states and of machines are not equivalence relations. The relations are reflexive and symmetric but they are not, in general, transitive.

Definition 3

A set of states, $S = \{s_1, s_2, \cdots, s_k\}$ of a machine M is called a set of *pseudoequivalent* states if and only if states s_i and s_j of S are compatible for all i and j, *i.e.*, if and only if the relation of compatibility is an equivalence relation for the set S.

III. Examples

As an example, consider the machine M, having three inputs and three outputs, which is defined by the following connection matrix:

$$\begin{bmatrix} 0 & (a, \alpha) & (b, \gamma) \\ (b, \gamma) & 0 & (c, \beta) \\ (a, \gamma) & 0 & (c, \beta) \end{bmatrix}.$$

The state diagram is shown in Fig. 1. The absence of certain inputs in the rows of the connection matrix is assumed to result from restrictions on the inputs.

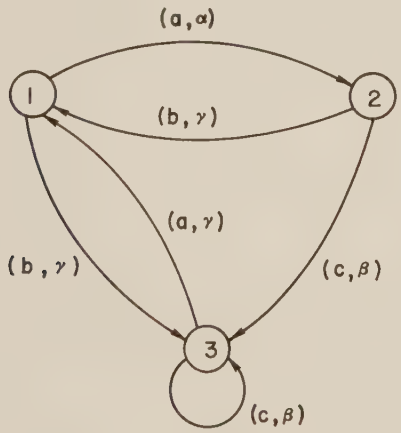


Fig. 1—State diagram of a machine in which there exist input restrictions.

For two states, i and j, to be compatible it is necessary that identical outputs be associated with each input common to rows i and j of the connection matrix. Therefore we see that states 1 and 3 cannot be compatible since identical outputs are not associated with the common input symbol a. States 1 and 2 as well as states 2 and 3 may be compatible. Examining states 2 and 3 we observe that all input sequences in the intersection of the two sets of associated allowable input sequences must start with the input symbol c. However, input c takes both states 2 and 3 into state 3 and gives the output β. Consequently, states 2 and 3 are compatible. States 1 and 2, on the other hand, are compatible if and only if states 1 and 3 are compatible, since input b takes the machine from state 1 to state 3 and from state 2 to state 1. Since states 1 and 3 are not compatible, neither are states 1 and 2. If the states of the machine are partitioned into the two sets, $S_1 = \{1\}$ and $S_2 = \{2, 3\}$ as indicated by the dashed lines in the connection matrix of M, then the states in each set are pseudoequivalent after definition 3. Letting the set $\{2, 3\}$ be represented by a single state $2'$ and the set $\{1\}$ by $1'$, we obtain the following connection matrix of a two-state machine M' which is compatible to the original machine under the correspondence $1 \leftrightarrow 1'$; $2, 3 \leftrightarrow 2'$:

$$\begin{bmatrix} 0 & (a, \alpha) \vee (b, \gamma) \\ (b, \gamma) \vee (a, \gamma) & (c, \beta) \end{bmatrix}.$$

The corresponding state diagram is given in Fig. 2.

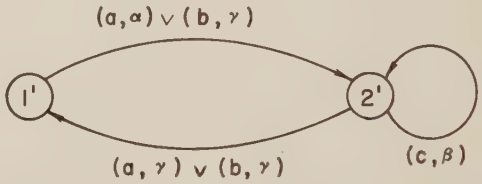


Fig. 2—State diagram of a machine compatible to the one defined in Fig. 1.

That the two machines are compatible can be ascertained in this example by comparing the input-output behavior of the two machines (a more formal treatment appears later). In other words, as long as only the originally allowable input sequences are presented to the two-state machine given in Fig. 2 it can be represented as the three-state machine described in Fig. 1. The connection matrix of this latter machine is readily obtained by replacing each submatrix of the partitioned connection matrix of machine M by the union of all its entries, a rule which will be formalized in Section VI.

It is important to note that the connection matrix for M' no longer contains all the information concerning the input restrictions of M. This information is given only in the original state diagram and the associated connection matrix. That the machine described in Fig. 1 could be represented by the compatible machine in Fig. 2 depended upon the existence of a certain partitioning of the connection matrix as illustrated in the connection matrix M. A detailed development of this observation is given below.

As a second example, consider the machine N, having 4 inputs and 4 outputs, defined by the following connection matrix.

$$\left[\begin{array}{ccc|cc} 0 & (c,\alpha) & 0 & 0 & (d,\gamma) \\ 0 & 0 & (c,\alpha) & 0 & 0 \\ 0 & (c,\alpha) & 0 & 0 & 0 \\ \hline (d,\gamma) & 0 & 0 & 0 & 0 \\ 0 & 0 & (c,\beta) & 0 & 0 \end{array}\right].$$

The associated state diagram appears in Fig. 3. From an inspection of the connection matrix we see that states 1, 2, and 3 form a set of pseudoequivalent states as does the set consisting of states 4 and 5. Indeed, the only common allowable sequence in the first case is $ccc\cdots$. In the second case there is none. Therefore, under the correspondence $1'\leftrightarrow 1, 2, 3$; $2'\leftrightarrow 4, 5$, a compatible machine N' which has only two states is obtained. The connection matrix of N', which is given below, can be obtained as in the previous example.

$$\left[\begin{array}{cc} (c,\alpha) & (d,\gamma) \\ (c,\beta)\vee(d,\gamma) & 0 \end{array}\right].$$

The state diagram is in Fig. 4.

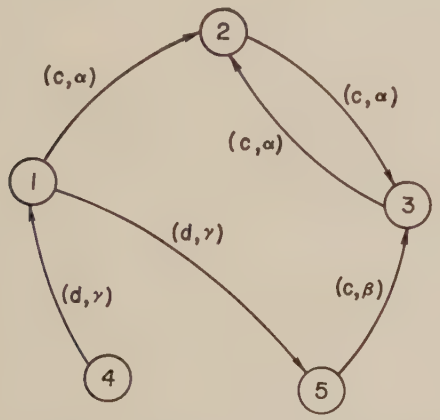


Fig. 3—Example of a machine in which there is more than one way to combine states under compatibility of states.

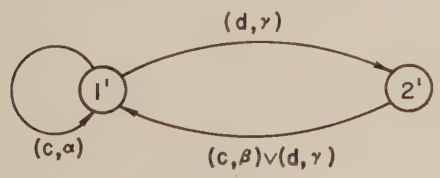


Fig. 4—A two-state machine compatible to the one defined in Fig. 3.

The above partitioning of states into pseudoequivalent sets is not the only one possible in this case. States 2, 3, and 4 also form a set of pseudoequivalent states since there is no common allowable input sequence. States 1 and 5 are not compatible since the input c is associated with a different output in the two cases. We have then as another partitioning the sets $S_1=\{1\}$, $S_2=\{2, 3, 4\}$, $S_3=\{5\}$. (S_1 and S_2 cannot be combined since state 1 is not compatible to state 4.) Partitioning the connection matrix with respect to the sets S_1, S_2, and S_3 gives

$$\left[\begin{array}{c|ccc|c} 0 & (c,\alpha) & 0 & 0 & (d,\gamma) \\ \hline 0 & 0 & (c,\alpha) & 0 & 0 \\ 0 & (c,\alpha) & 0 & 0 & 0 \\ (d,\gamma) & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & (c,\beta) & 0 & 0 \end{array}\right].$$

The connection matrix of a compatible 3 state machine, N'', under the correspondence $1'\leftrightarrow 1$; $2'\leftrightarrow 2, 3, 4$; $3'\leftrightarrow 5$ is given below while the state diagram appears in Fig. 5.

$$\left[\begin{array}{ccc} 0 & (c,\alpha) & (d,\gamma) \\ (d,\gamma) & (c,\alpha) & 0 \\ 0 & (c,\beta) & 0 \end{array}\right].$$

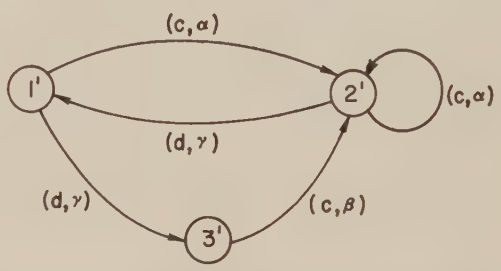


Fig. 5—A three-state machine compatible to the one defined in Fig. 3, but not compatible to the machine described in Fig. 4.

A check of the connection matrix of N'' reveals that no further nontrivial partitioning under pseudoequivalence exists. Since machines N' and N'' do not have the same number of states we see that different simplifications of a given state diagram with respect to pseudoequivalence do not necessarily lead to representations involving the same number of states.

We also note that although machines N' and N are compatible, as are machines N and N'', machine N' is not compatible to machine N''. This illustrates the assertion that compatibility of machines is not necessarily transitive.

To verify the noncompatibility of machines N' and N'', observe that the input sequence c, d, c presented to machine N' when it is in state $1'$ produces the output sequence α, γ, β. If the same input sequence is applied to machine N'' the above sequence of outputs is not obtained under any starting condition.

The allowable input sequences of a machine M are, as has been noted,[8] obtainable from the powers of the connection matrix C. The same holds true for the connection matrix C' of an arbitrary reduced (with respect

to compatibility of states) form M' of M. If a state s_i of M maps into a state t_j of M', then the allowable input sequences of length r associated with s_i all appear in row j of $C'(r)$, along with any additional input sequences of length r which may be allowable for state t_j of M' by virtue of the combination of states of M. It is significant that allowable sequences of length r may well appear in association with states of M' whereas these sequences are allowable for no state of the original machine. This re-emphasizes the fact that a reduced machine M' is not ordinarily equivalent to the original machine M, but rather performs the same as M only for sequences allowable for M.

IV. Fundamental Theorems

Definition

A matrix A whose elements are input-output polynomials[8] is called a *generalized r matrix* if it has the following properties:

1) All nonzero entries of A are homogeneous and of a common degree r,
2) All nonzero terms in each row of A have distinct input sequences, and,
3) If an input sequence appears in more than one row of the matrix all associated output sequences are identical. (If all input sequences which appear in any given row also appear in every other row, then a generalized r matrix becomes an r matrix.)[10]

Lemma 1: The union[8] of two generalized r matrices A and B of the same order is again a generalized r matrix whenever the terms of corresponding rows of A and B contain no identical input sequences.

Lemma 2: The product[8] AB of a generalized r matrix A and a generalized s matrix B is a generalized $(r+s)$ matrix.

Theorem 1: Let the submatrices of a symmetric partitioning of a given connection matrix C be denoted by C_{ij}, $i, j=1, 2, \cdots, q$. Let the submatrices of an identical partitioning of C^r, where r is any positive integer, be denoted by $C_{ij}^{(r)}$, $i, j=1, 2, \cdots, q$. If now each C_{ij} is a generalized 1 matrix and if in each row of submatrices $C_{i1} \cdots C_{iq}$ no two distinct submatrices have any common identical inputs, then each $C_{ij}^{(r)}$ is a generalized r matrix and in each row of submatrices $C_{i1}^{(r)} \cdots C_{iq}^{(r)}$ no two distinct submatrices have any input sequences in common.

Theorem 2: If a connection matrix admits a symmetrical partitioning in which each submatrix is a generalized 1 matrix and in which no two distinct submatrices in any given row of submatrices have an input symbol in common, then the states contained in each set of the corresponding partitioning of states are pseudoequivalent.

The proofs are similar to the proofs of the corresponding lemmas and theorems in the paper on equivalence of states.[11]

V. Algorithm for Determining Partitionings into Sets of Pseudoequivalent States

Consider the connection matrix C of any machine M. The following procedure will determine partitionings into sets of pseudoequivalent states. As we have already noted in Section III, the partitioning of the states of a machine into sets of pseudoequivalent states is not necessarily unique; therefore the number of sets in the resulting partitioning will, ordinarily, depend upon how the partitioning was obtained.

1) Partition the states of M into disjunct sets S_1, $S_2, \cdots, S_q$ such that the rows of C associated with the states in each S_i form a generalized 1 matrix and such that this condition is not preserved if any two sets S_i, S_k are combined. Note that this partitioning is not uniquely defined. Ordinarily, there will be more than one such partitioning.

If the only partitioning is the trivial partitioning (into sets of order 1), then there are no compatible, and hence no nontrivial, sets of pseudoequivalent states in M.

2) If there is a nontrivial partitioning $S_1, S_2, \cdots, S_q$, reorder the states according to this partitioning, if necessary, so that the rows corresponding to states in the same S_i are adjacent in the matrix, and hence it is possible to partition the corresponding connection matrix C symmetrically with respect to the S_i into generalized 1 matrices. Indicate the partitioning by $C=[C_{ij}]$.

If the C_{ij} are all generalized 1 matrices so that for all j and k C_{ij} and C_{ik} have no nonzero common inputs where C_{ij} and C_{ik} are any two submatrices in the ith row of submatrices, then the process terminates at this point because the states in each S_i are pseudoequivalent by Theorem 2.

3) If the C_{ij} are not all generalized 1 matrices so that C_{ij} and C_{ik} have no nonzero inputs in common, then partition the states associated with each row of submatrices $C_{i1}, C_{i2}, \cdots, C_{iq}$, $i=1, 2, \cdots, q$, into disjunct sets $S_{i1}, S_{i2}, \cdots, S_{ih_i}$ so that for a given S_{ij} each corresponding set of rows in $C_{i1}, C_{i2}, \cdots, C_{iq}$ is a generalized 1 matrix; so that no two of these associated submatrices corresponding to a given S_{ij} have any nonzero inputs in common; and finally so that these conditions are not preserved if any two sets S_{ij}, S_{ik} are combined. Again, there will be, in many cases, more than one such partitioning. If the only partitioning is the

[10] *Ibid.*, p. 280.

[11] *Ibid.*, p. 281.

trivial one for all rows of submatrices $C_{i1}, C_{i2}, \cdots, C_{iq}$, $i=1, 2, \cdots, q$, and for all partitionings indicated in step 1, then there are no compatible states.

4) If there is a nontrivial partitioning, $S_{i1}, S_{i2}, \cdots, S_{ih_i}$, resulting from step 3, then repartition the connection matrix with respect to the S_{ij}, again reordering the states, if necessary, so as to have the rows of states in the same set S_{ij} adjacent to each other in the matrix. Denote the new partitioning by

$$C = [C_{ij}] = [D_{ij}^{kt}]; \quad C_{ij} = [D_{ij}^{kt}]_{(ij \text{ fixed})},$$

where the indexes k and t are used to denote the submatrices in the partitioning of each submatrix C_{ij}.

If now the D_{ij}^{kt} are all generalized 1 matrices so that for all t and s D_{ij}^{kt} and D_{ij}^{ks} have no nonzero inputs in common, the process terminates since the states in each S_{ij} form a set of pseudoequivalent states after Theorem 2. Otherwise, steps 3 and 4 are repeated.

We have thus outlined a two-step process: first, partition the sets of states into disjunct sets so that the submatrices of the previous partitioning are generalized 1 matrices with no two of the newly formed submatrices in the same row of submatrices having a nonzero input in common, and so that these conditions are not preserved if any two of these newly formed sets of states are combined; second, repartition the connection matrix with respect to this new partitioning of states. These two steps are repeated until a partitioning of the connection matrix is obtained in which *all* submatrices are generalized 1 matrices with no two submatrices in the same row having an input in common. This is a finite process since the number of states is finite. When such a partitioning is obtained, then by Theorem 2 we see that all states in each subset of states in the final partitioning form a set of pseudoequivalent states. If the subsets of states ultimately obtained are all of order 1 for *all possible* partitionings indicated in the second step, then there are no compatible states, and thus no nontrivial sets of pseudoequivalent states.

We have already noted that the number of sets of pseudoequivalent states ultimately obtained depends upon how the partitioning was effected at each stage. An unsolved problem of considerable interest is to determine the minimum number of sets of pseudoequivalent states without repeating the above process for each possibility and comparing results.

VI. The Reduced Machine Under Pseudoequivalence

After completing the process outlined in Section V, we define a new machine by replacing, in effect, the states of each pseudoequivalent set by a single state representing that set. To account for the transitions of all the states of one set to states of other sets under the various inputs to the machine M, we replace each submatric of the final partitioned form C by the union of all its entries. For simplicity of notation denote the final partitioned form of C by

$$C = [C_{ij}], \qquad i, j = 1, 2, \cdots, q,$$

and the corresponding sets of pseudoequivalent states by $S_1, S_2, \cdots, S_q$. Then if d_{ij} is the union of all the entries of C_{ij}, the $q \times q$ matrix

$$D = [d_{ij}]$$

is the connection matrix of a machine N with q states, which is compatible to M. That D is indeed a connection matrix is seen by noting that because C is a connection matrix, the C_{ij}'s are generalized 1 matrices, and no two matrices C_{ij} and C_{ik} in the same row of submatrices contain any common nonzero input symbols. Therefore, d_{ij} and d_{ik} will contain no common nonzero input symbol, which is the only condition to be satisfied.

The machine N defined by D is called a *reduced form* of M under pseudoequivalence. (It may very well be the case that there exist compatible states in N and even sets of pseudoequivalent states.) It remains to show that M and N are compatible machines. To prove this, for each positive integer r partition C^r according to the indicated sets of pseudoequivalent states of M, and let $C_{ij}^{(r)}$ denote any of the resulting submatrices. The entries of $C_{ij}^{(r)}$ define the allowable input sequences of length r which take the states of set S_i into those of set S_j and also give the associated output sequences.

Now the entry d_{ij} of D is the union of the entries C_{ij} and hence defines the set of all inputs which take some member of S_i into some member of S_j. On the other hand, $D_{ij}^{(r)}$, the ij entry of D^r, is given by

$$d_{ij}^{(r)} = \bigvee_{k_1, k_2, \cdots, k_{r-1}=1}^{r} d_{ik_1} d_{k_1 k_2} \cdots d_{k_{r-1} j}.$$

Since each state of M appears in some S_i, every term appearing in $C_{ij}^{(r)}$ has its factors appearing in some product of the form $d_{ik_1} d_{k_1 k_2} \cdots d_{k_{r-1} j}$, and hence is a term of $d_{ij}^{(r)}$.

From these observations we see that all the allowable input sequences associated with each of the states of M in S_i are also allowable input sequences for state i of N (the converse is not necessarily the case), and the output sequences associated with a common input sequence are identical. Hence, by Definition 2 of Section II, the machines M and N are compatible. We then have the following theorem.

Theorem 3: Given a machine M, there exists a corresponding compatible machine N (not necessarily unique) with minimum numbers of states.

At this point, the minimum number of states can be determined only by examining all possibilities.

VII. An Example

In order to illustrate the algorithm in Section IV we consider a machine having 6 inputs—a, b, c, d, e, f; 4 outputs —$\alpha, \beta, \gamma, \delta$; and 13 states, which is defined by the following connection matrix:

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	(e, δ)	0	0	0	0	0	0	$(a, \alpha) \vee (c, \beta)$	0	0	0	0
2	0	(a, β)	(e, β)	0	0	0	0	0	(c, β)	(d, α)	(b, γ)	0	0
3	(e, δ)	0	(b, γ)	0	0	0	0	0	(a, α)	0	(c, β)	0	0
4	(a, α)	0	0	(b, γ)	0	0	0	0	(e, δ)	0	(c, β)	0	0
5	0	0	(d, α)	(e, β)	(b, γ)	0	(a, β)	0	0	0	0	0	(c, β)
6	(e, α)	0	(f, γ)	0	0	(a, β)	0	0	0	(b, δ)	0	(d, α)	0
7	$(c, \beta) \vee (e, \delta)$	(b, γ)	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	(d, α)	0	0	0	(b, γ)	0	0
9	0	0	(b, γ)	0	0	0	0	(e, δ)	0	0	0	0	0
10	0	0	0	0	0	0	0	(b, γ)	(c, β)	(a, α)	0	0	0
11	(c, β)	$(b, \gamma) \vee (a, \beta)$	0	0	0	0	(d, α)	0	0	0	0	(e, β)	0
12	0	(e, δ)	0	(b, γ)	0	0	0	0	0	0	0	(a, α)	0
13	0	0	(f, γ)	0	0	0	(b, δ)	0	(d, α)	0	0	(e, α)	(a, β)

There are two distinct, disjunct partitionings so that the corresponding sets of rows form generalized 1 matrices and so that this condition is not preserved if any two sets of states are combined: P_1: $\{1, 3, 4, 10, 12, 7, 9\}$, $\{2, 5, 11, 8\}$, $\{6, 13\}$; P_2: $\{1, 3, 4, 7, 9, 12, 10, 8\}$, $\{2, 5, 11\}$, $\{6, 13\}$. If the above connection matrix is partitioned with respect to P_1, we have

	1	3	4	7	9	10	12	8	2	5	11	6	13
1	0	0	0	0	$(a, \alpha) \vee (c, \beta)$	0	0	0	(e, δ)	0	0	0	0
3	(e, δ)	(b, γ)	0	0	(a, α)	0	0	0	0	0	(c, β)	0	0
4	(a, α)	0	(b, γ)	0	(e, δ)	0	0	0	0	0	(c, β)	0	0
7	$(c, \beta) \vee (e, \delta)$	0	0	0	0	0	0	0	(b, γ)	0	0	0	0
9	0	(b, γ)	0	0	0	0	0	(e, γ)	0	0	0	0	0
10	0	0	0	0	(c, β)	(a, α)	0	(b, γ)	0	0	0	0	0
12	0	0	(b, γ)	0	0	0	(a, α)	0	(e, δ)	0	0	0	0
8	0	0	0	(d, α)	0	0	0	0	0	0	(b, γ)	0	0
2	0	(e, β)	0	0	(c, β)	(d, α)	0	0	(a, β)	0	(b, γ)	0	0
5	0	(d, α)	(e, β)	(a, β)	0	0	0	0	0	(b, γ)	0	0	(c, β)
11	(c, β)	0	0	(d, α)	0	0	(e, β)	0	$(b, \gamma) \vee (a, \beta)$	0	0	0	0
6	(e, α)	(f, γ)	0	0	0	(b, δ)	(d, α)	0	0	0	0	(a, β)	0
13	0	(f, γ)	0	(b, δ)	(d, α)	0	(e, α)	0	0	0	0	0	(a, β)

We note that the rows of submatrices corresponding to the two sets, $\{1, 3, 4, 7, 9, 10, 12\}$ and $\{8, 2, 5, 11\}$ are not all generalized 1 matrices so that any two submatrices in the same row have no common nonzero input symbols. Therefore, the states in these two sets must be partitioned in accordance with the algorithm. The only two distinct, nonsuperfluous disjunct partitionings are:

P_{21}: $\{1, 9, 12\}$, $\{3, 4\}$, $\{7, 10\}$, $\{2, 11, 8\}$, $\{5\}$, $\{6, 13\}$; P_{22}: $\{1, 9, 12\}$ $\{3, 4\}$, $\{7, 10\}$, $\{2, 11\}$, $\{8, 5\}$, $\{6, 13\}$. The partitioning P_{21} leads to the following connection matrix:

	1	9	12	3	4	7	10	2	8	11	5	6	13
1	0	$(a,\alpha) \vee (c,\beta)$	0	0	0	0	0	(e,δ)	0	0	0	0	0
9	0	0	0	(b,γ)	0	0	0	0	(e,δ)	0	0	0	0
12	0	0	(a,α)	0	(b,γ)	0	0	(e,δ)	0	0	0	0	0
3	(e,δ)	(a,α)	0	(b,γ)	0	0	0	0	0	(c,β)	0	0	0
4	(a,α)	(e,δ)	0	0	(b,γ)	0	0	0	0	(c,β)	0	0	0
7	$(c,\beta) \vee (e,\delta)$	0	0	0	0	0	0	(b,γ)	0	0	0	0	0
10	0	(c,β)	0	0	0	0	(a,α)	0	(b,γ)	0	0	0	0
2	0	(c,β)	0	(e,β)	0	0	(d,α)	(a,β)	0	(b,γ)	0	0	0
8	0	0	0	0	0	(d,α)	0	0	0	(b,γ)	0	0	0
11	(c,β)	0	(e,β)	0	0	(d,α)	0	$(b,\gamma) \vee (a,\beta)$	0	0	0	0	0
5	0	0	0	(d,α)	(e,β)	(a,β)	0	0	0	0	(b,γ)	0	(c,β)
6	(e,α)	0	(d,α)	(f,γ)	0	0	(b,δ)	0	0	0	0	(a,β)	0
13	0	(d,α)	(e,α)	(f,γ)	0	(b,δ)	0	0	0	0	0	0	(a,β)

The row of submatrices corresponding to set of states $\{2, 8, 11\}$ are not generalized 1 matrices so that no two submatrices have a common nonzero input symbol. There are again two possible partitionings: P_{211}: $\{1, 9, 12\}$, $\{3, 4\}$, $\{7, 10\}$, $\{2, 8\}$, $\{11\}$, $\{6, 13\}$; P_{212}: $\{1, 9, 12\}$, $\{3, 4\}$, $\{7, 10\}$, $\{2\}$ $\{8, 11\}$, $\{5\}$, $\{6, 13\}$. The partitioning P_{211} results in the following connection matrix:

	1	9	12	3	4	7	10	2	8	11	5	6	13
1	0	$(a,\alpha) \vee (c,\beta)$	0	0	0	0	0	(e,δ)	0	0	0	0	0
9	0	0	0	(b,γ)	0	0	0	0	(e,δ)	0	0	0	0
12	0	0	(a,α)	0	(b,γ)	0	0	(e,δ)	0	0	0	0	0
3	(e,δ)	(a,α)	0	(b,γ)	0	0	0	0	0	(c,β)	0	0	0
4	(a,α)	(e,δ)	0	0	(b,γ)	0	0	0	0	(c,β)	0	0	0
7	$(c,\beta) \vee (e,\delta)$	0	0	0	0	0	0	(b,γ)	0	0	0	0	0
10	0	(c,β)	0	0	0	0	(a,α)	0	(b,γ)	0	0	0	0
2	0	(c,β)	0	(e,β)	0	0	(d,α)	(a,β)	0	(b,γ)	0	0	0
8	0	0	0	0	0	(d,α)	0	0	0	(b,γ)	0	0	0
11	(c,β)	0	(e,β)	0	0	(d,α)	0	$(b,\gamma) \vee (a,\beta)$	0	0	0	0	0
5	0	0	0	(d,α)	(e,β)	(a,β)	0	0	0	0	(b,γ)	0	(c,β)
6	(e,α)	0	(d,α)	(f,γ)	0	0	(b,δ)	0	0	0	0	(a,β)	0
13	0	(d,α)	(e,α)	(f,γ)	0	(b,δ)	0	0	0	0	0	0	(a,β)

We observe that all submatrices are now generalized 1 matrices with no two submatrices in the same row having nonzero input symbols in common. Thus, each set of states in the corresponding partitioning of the states forms a set of pseudoequivalent states. The reduced form of the machine relative to this particular partitioning into pseudoequivalent states is given by the following connection matrix:

	1	3	7	2	11	5	6
1	$(a, \alpha) \vee (c, \beta)$	(b, γ)	0	(e, δ)	0	0	0
3	$(a, \alpha) \vee (e, \delta)$	(b, γ)	0	0	(c, β)	0	0
7	$(c, \beta) \vee (e, \delta)$	0	(a, α)	(b, γ)	0	0	0
2	(c, β)	(e, β)	(d, α)	(a, β)	(b, γ)	0	0
11	$(c, \beta) \vee (e, \beta)$	0	(d, α)	$(b, \gamma) \vee (a, \beta)$	0	0	0
5	0	$(d, \alpha) \vee (e, \beta)$	(a, β)	0	0	(b, γ)	(c, β)
6	$(d, \alpha) \vee (e, \alpha)$	(f, γ)	(b, δ)	0	0	0	(a, β)

There are no compatible states in the above connection matrix.

We do not include a corresponding development for the other possible partitionings. P_{212}, for example, leads to the trivial partitioning. P_1 gives rise to a number of different decompositions into sets of pseudoequivalent states, none, leading however, to a compatible machine with fewer states than the machine resulting from P_{211}.

VIII. Conclusion

The relation of compatibility of states and of machines permits a simplification of state diagrams in addition to that provided by the relation of equivalence of states and of machines. A method has been developed for reducing a given state diagram to its "simplest" compatible form.

IX. Acknowledgment

The author wishes to thank Professor Franz Hohn for a number of helpful discussions in connection with the above development.

Theoretical Consideration of Computing Errors of a Slow Type Electronic Analog Computer*

T. MIURA† and M. NAGATA†

Summary—**When solving differential equations using analog computers, there are two causes of errors: one results from the integrators and the other results from the coefficient setting elements. The former is independent of the order of the differential equation if the integrating time constants are all equal, while the latter is dependent on the setup procedure. The amount of the error from the coefficient setting elements is comparatively small in the operating frequency range from $\omega \leq$rad/sec. Therefore, theoretical analysis considering errors due to the integrators only is quite sufficient. However, higher accuracy is required in $\omega = 10$ to 100 rad/sec (for example in flight simulators) and consequently the error caused by the coefficient setting elements has become impossible to neglect.**

The authors have succeeded in formulating a generalized and practical equation developed from conventional formulas.

Introduction

THERE have been several reports, including Macnee's, of analyses of errors caused by solving linear differential equations by means of electronic analog computers. However, from the standpoint of practical usage, the analyzed computer setups in all these reports are not general; for instance, Macnee[1] treated the case in which all the constant coefficient multipliers are used to set coefficients without considering the sign of the operational amplifiers and in which the constant coefficient multipliers are fed back through the adder to the integrator corresponding to the highest order of the differential equation. Marsocci[2] treated the case where the potentiometers and the sign changers are used as coefficient setting elements and the sign changers are used in order to align the sign. A further condition was that every coefficient k_m was in the range $0 \leq k_m \leq 1$. Kurokawa[3] treated the computer setup excluding the constant coefficient multiplier used in Macnee's case,

* Manuscript received by the PGEC, June 25, 1958.
† Hitachi Central Res. Lab., Tokyo, Japan.

[1] A. B. Macnee, "Some limitation on the accuracy of electronic differential analyzers," Proc. IRE, vol. 40, pp. 303–308; March, 1952.
[2] V. A. Marsocci, "An error analysis of electronic analog computers," IRE Trans. on Electronic Computers, vol. 5, pp. 207–212; April, 1956.
[3] K. Kurokawa, "An Error Evaluation of Electronic Analogue Computers," preparatory essay of the seminar of Japan Sci., Assn.; 1957. (In Japanese.)

while Nomura[4] pointed out that a part of Macnee's analysis was incorrect and showed the required correction.

In all the above cases except Nomura's, the frequency characteristics are given in the form of the operational amplifier (closed-loop characteristics). The authors[5–7] already have introduced a practical and general equation of the frequency characteristics in the form of the dc amplifier (open-loop frequency characteristics) on the assumption that errors caused by the constant coefficient multipliers and the adders may be assumed to be negligibly small compared with the errors of the integrators.[8]

However, recently higher accuracy in the operating frequency range of $\omega \geq 10$ rad/sec (such as in flight simulators) has been required, and it is very often necessary to connect constant coefficient multipliers in tandem. Consequently it is impossible to neglect the errors due to the constant coefficient multipliers and the errors due to the associated stray capacities. Therefore it is essential to develop general formulas considering the errors of the coefficient setting elements.

This paper discusses theoretically the errors due to computer setups, including constant coefficient multipliers and potentiometers, *etc.* In the following discussions, computing impedances are assumed to have proper values.

Analyzed Computer Setup

The circuit shown in Fig. 1 is a computer setup for solving a linear differential equation with constant coefficients, and it is used in this analysis.

The newly-developed formula has the following features: the general expression of the transfer function $Km(p)$ for all coefficient setting elements is applied for all the cases of potentiometer or constant coefficient multiplier settings and the sign of the integrator is included. This also is true for all the other computing elements, and errors due to the adders are included in the sign changer as a first order approximation. The differential equation which is solved by the computer shown in Fig. 1 is

$$\sum_{m=0}^{n} T^m p^m k_m y = 0 \quad \text{where} \quad k_n = 1. \tag{1}$$

Now, considering the transfer function $I(p)$ of the integrator and the transfer function $Km(p)$ of the coefficient setting element in Fig. 1, we have

$$I(p)^{-1} = \{1 + (1 + \mu_i) Z_i(p) Y(p)\}/\mu_i, \tag{2}$$

where $Z_i(p)$, $Y(p)$ are the equivalent input computing impedance and the feedback computing admittance respectively. (In the following considerations we neglect the effect of the grid current and the drift voltage of the integrator, which will be treated in Appendix I.) The following relationships are obtained if we take into consideration the inductance and the stray capacitance across the integrating resistor, and the leakage resistance and the dielectric absorption across the integrating capacitor in Fig. 2. (For the dielectric absorption phenomenon, Cole-Cole's experimental formula[6] which gives a good approximation for high molecule materials was applied.)

$$Z_i(p) \simeq R\{1 - C_s R p + (L/R)p\}$$

$$Y_i(p) \simeq C\left(p + \frac{C_1}{C}\,\frac{p^\alpha}{\tau_\epsilon^{1-\alpha}} + \frac{1}{CR_l}\right). \tag{3}$$

From (2) and (3)

$$I(p)^{-1} = T\left[\frac{1}{\mu_i T} + \left\{p + \frac{p}{\mu_i} + \Delta(p)\right\}\right]$$

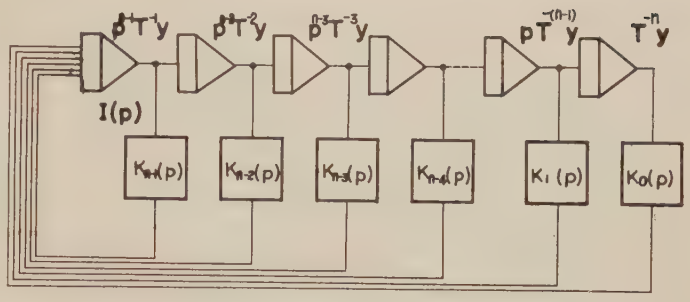


Fig. 1—Computer setup for solving equation (1).

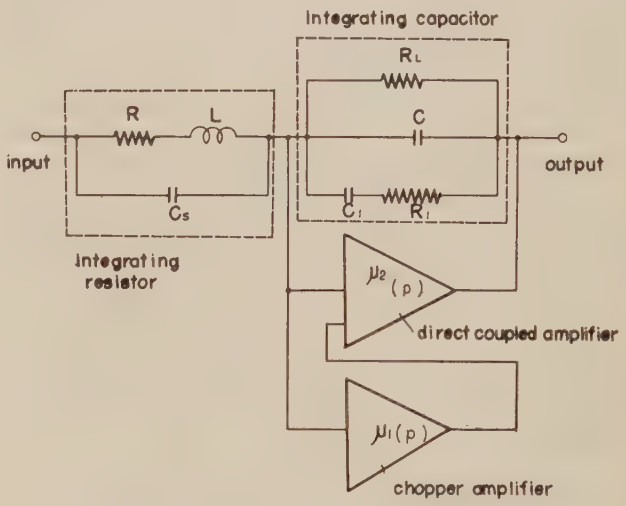


Fig. 2—Equivalent circuit of integrator.

[4] T. Nomura, "An Error of Electronic Analog Computer," Industrial Lab., Tokyo University, Tokyo, Japan, Rep. No. 414; 1953. (In Japanese.)

[5] T. Miura, "The electronic analog computer," *Hitachi Rev.*, (special issue for electronic tube and application of electronic tube) special issue No. 3, pp. 139–145; 1953. (In Japanese.)

[6] T. Miura, M. Aoki, M. Hibi, and T. Numakura, "On the precision of a slow type analog computer," *J. IEE Japan*, vol. 76, pp. 815, 896–905; August, 1956. (In Japanese.)

[7] T. Mirua, Z. Abe and M. Nagata, "On Computing Errors of a Slow Type Analog Computer," essays presented at IEE Conference, Tokyo, Japan; November, 1957. (In Japanese.)

[8] In practice it is possible to neglect the errors due to the integrators when operating in the frequency range of $\omega \leq 10$ rad/sec.

where

$$\Delta(p) = -C_s R p^2 + (L/R)p^2 + \frac{C_1}{C}\frac{p^\alpha}{\tau_\epsilon^{1-\alpha}} + \frac{1}{CR_i}. \quad (4)$$

The general formula for $Km(p)$ is

$$K_m(p) = k_m\{1 - (K_m/\mu_k) + p\tau_m\}. \quad (5)$$

The second and third terms of the right-hand side of (5) show the error terms caused by amplifier gain μ_k and the stray capacitance across the computing resistors respectively. The determination of Km, τ_m, can be seen by referring to Table I.

If we used the transfer function for each of the computing elements, the characteristic equation for the case of computer setup of Fig. 1 is

$$\sum_{m=0}^{n} I(p)^{-m} K_m(p) \equiv F(p) = 0. \quad (6)$$

Assuming that the characteristics of the operational amplifier are ideal, then from (4) and (5) we have

$$I(p)^{-1} = T \cdot p, \qquad K_m(p) = k_m. \quad (7)$$

Substituting (7) into (6), we find (6) is identical to (1). We will consider the significance of (6) in the following section.

Introduction of the General Formula for Over-all Computing Errors

It may be adequate to think of the quantity ϵ for examining the error which is given by the difference between one root p_a' obtained from the characteristic (6) and the characteristic root p_a from (1). By applying Taylor's expansion theorem, (6) becomes

$$F(p_a') = F(p_a) + \epsilon\left[\frac{dF(p)}{dp}\right]_{p_a} + \frac{\epsilon^2}{2}\left[\frac{d^2F(p)}{dp^2}\right]_{p_a} + \cdots + \frac{\epsilon^n}{n!}\left[\frac{d^nF(p)}{dp^n}\right]_{p_a} + \cdots.$$

In general $d^nF(p)/dp^n$ has a finite value and the above series converges for any value of ϵ. Furthermore, comparing the second and the third terms, obviously the latter is smaller than the former at the computing frequencies in which case we may assume $1 \gg |\epsilon/p|$ to be correct. Therefore we can neglect the third and the following higher order terms.

$$\epsilon = -F(p_a)/[dF(p)/dp]_{p_a}. \quad (8)$$

Calculating $dF(p)/dp$ from (6)

$$\frac{dF(p)}{dp} = -I(p)^{-2}\frac{dI(p)}{dp}\left[\sum_{m=1}^{n} mI(p)^{-(m-1)}K_m(p)\right] + \sum_{m=0}^{n-1}\frac{dK_m(p)}{dp}I(p)^{-m}. \quad (9)$$

The term $-I(p)^{-2}\cdot dI(p)/dp$, $dKm(p)/dp$ within (9) can be reduced to

$$-I(p)^{-2}\frac{dI(p)}{dp} \simeq T\{1 + (1/\mu_i) + (d\Delta(p)/dp)\}$$
$$\frac{dK_m(p)}{dp} \simeq \tau_m \cdot k_m. \quad (10)$$

Substituting (10) which has been obtained by the first approximation of (9) gives

$$\frac{dF(p)}{dp} \simeq \sum_{m=1}^{n} mT^m k_m\left\{1 + \frac{1}{\mu_i} + \frac{d\Delta(p)}{dp} + (m-1)\left(\frac{1}{\mu_i} + \frac{\Delta(p)}{p}\right) - \frac{K_m}{\mu_K} - p\tau_m\right\}p^{m-1} + \frac{1}{\mu_i}\sum_{m=1}^{n} m(m-1)T^{m-1}p^{m-2}k_m + \sum_{m=0}^{n-1}\tau_m k_m T^m p^m. \quad (11)$$

[9] We write $p_a = p$ hereafter unless otherwise stated.

TABLE I
The Method of Determination of $Km(p)$ and τ_m

(a) Determination of $Km(p)$	
1) Coefficient setting potentiometer.	$Km=0$
2) The tandem connection of N constant coefficient multipliers.	$Km=\sum_{m=1}^{n}(1+Cm)$* where Cm is the multiplication of each constant coefficient multiplier; *i.e.*, for sign changer, $Km=1+1=2$. For the tandem connection of constant coefficient multipliers with multiplication 5 and 6, $Km=6+7=13$.
3) (Coefficient setting potentiometer)+(constant coefficient multiplier).	The same as 2).
4) The case through the adder.	The determination of Km is possible from 2) on the assumption that passing through one sign changer is equal to passing through one adder.
(b) Determination of τ_m	
1) Constant coefficient multiplier.	$\tau_m=\tau_{im}-\tau_{fm}$ $C_{si}R_i=\tau_{im}$ $C_{sf}R_f=\tau_{fm}$ where R_i: input computing resistor R_f: feedback computing resistor C_{si}: stray capacitor across the input computing resistor C_{sf}: stray capacitor across the feedback computing resistor.
2) Tandem connection of constant coefficient multipliers.	$\sum \tau_m$
3) Coefficient setting potentiometer.	$\tau_m=T_1-T_2$ $T_1=(1-km)RC_1$ $T_2=km(1-km)R(C_1+C_2)$ C_1, $C_2\simeq 100\sim 300pF$ where R: resistance between A and B terminals C_1: stray capacitor between A and C terminals C_2: stray capacitor between C and B terminals km: dividing ratio A and B are each terminals of the coefficient setting potentiometer and C is the slider.

* See Appendix II.

In case the first approximation is possible for ϵ, $dF(p)/dp$ can be approximated by the following:

$$\frac{dF(p)}{dp} \simeq \sum_{m=1}^{n} mT^m k_m p^{m-1}. \tag{12}$$

Next putting (4) and (5) into (6), then

$$\begin{aligned} F(p) &= \sum_{m=0}^{n} T^m p^m \left\{1 + \frac{m}{\mu_i} + m\frac{\Delta(p)}{p}\right\}\left(1 - \frac{K_m}{\mu_K} + p\tau_m\right) k_m \\ &\quad + \frac{1}{\mu_i T}\sum_{m=1}^{n} mT^m p^{m-1}\left\{1 + \frac{m-1}{\mu_1} \right. \\ &\quad \left. + (m-1)\frac{\Delta(p)}{p}\right\} k_m\left(1 - \frac{K_m}{\mu_K} + p\tau_m\right) \\ &\simeq \sum_{m=0}^{n} T^m p^m k_m \left\{1 + \frac{m}{\mu_i} + m\frac{\Delta(p)}{p} - \frac{K_m}{\mu_K} + p\tau_m\right\} \\ &\quad + \frac{1}{\mu_i T}\sum_{m=1}^{n} mT^m p^{m-1} k_m. \end{aligned} \tag{13}$$

Substituting (1) and (12) into (13)

$$\begin{aligned} F(p) &= \sum_{m=0}^{n} T^m p^m k_m \left\{\frac{m}{\mu_i} + m\frac{\Delta(p)}{p} - \frac{K_m}{\mu_K} + p\tau_m\right\} \\ &\quad + \frac{1}{\mu_i T}\frac{dF(p)}{dp}. \end{aligned} \tag{14}$$

From (8), (13) and (14)

$$\begin{aligned} \epsilon = &-\frac{1}{\mu_i T} - \Delta(p) - \frac{p}{\mu_i} + \frac{1}{\mu_K}\frac{\sum_{m=0}^{n} T^m p^m k_m K_m}{\sum_{m=1}^{n} mT^m p^{m-1} k_m} \\ &- \frac{\sum_{m=0}^{n} T^m p^{m+1} k_m \tau_m}{\sum_{m=1}^{n} mT^m p^{m-1} k_m} \end{aligned}$$

where

$$\Delta(p) = -C_s R p^2 + (L/R)p^2 + \frac{C_1}{C}\frac{p^\alpha}{\tau_\epsilon^{1-\alpha}} - \frac{1}{CR_l}, \qquad \tau_n = 0,\ K_n = 0. \tag{15}$$

Eq. (15) is the generalized equation of errors. The first and third terms of the right-hand side of (15) are the error terms caused by the dc amplifiers of the integrator, while the second term shows the errors caused by the computing impedance of the integrator, and the fourth and fifth terms show the errors caused by the dc amplifiers of the constant coefficient multiplier and the computing resistor respectively.

From (15) it is obvious that the error of the integrator is quite independent of the order of the differential equation in the case where the integrating time constants are all equal. Therefore, the estimation of the error of the integrator by the circle test (analysis of the second-order differential equation with no damping) is possible and, what is more, it may become the necessary condition of the determination of the over-all accuracy when using this test.

The approximating equation for ϵ, taking the effect of the frequency characteristics of the amplifiers into consideration, can be easily obtained from (15) by treating μ_i and μ_k as functions of p and considering the effects of their frequency characteristics. A combination type dc amplifier which has an automatic balancing circuit is used. Assuming that time constants, which provide the gain and the frequency characteristics of the chopper part, and the direct-coupled amplifier part are as shown in Fig. 3, then

$$\mu \simeq \frac{\mu_0(1 + p\tau_0)}{(1 + p\tau_1)(1 + p\tau_2)} \tag{16}$$

where $\mu_0 = \mu_1 \cdot \mu_2$, $\tau_0 = \tau_1/\mu_1$, $\mu_1 \gg 1$.

In (16) assuming $p = p_\alpha'$ and substituting this equation into (15), we can obtain by the method of successive approximations the following equation as a first approximation.

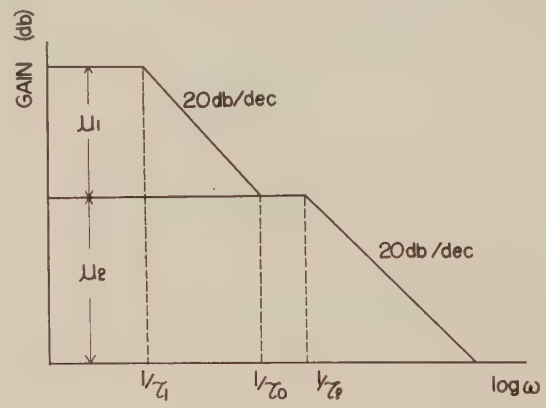


Fig. 3—Frequency characteristic of the operational amplifier.

$$\begin{aligned} \epsilon \simeq &-\frac{(1 + p\tau_{i1})(1 + p\tau_{i2})}{\mu_{0i}(1 + p\tau_{i0})}\left(p + \frac{1}{T}\right) - \Delta(p) \\ &+ \frac{(1 + p\tau_{K1})(1 + p\tau_{K2})}{\mu_{0K}(1 + p\tau_{K0})}\frac{\sum_{m=0}^{n} T^m p^m k_m K_m}{\sum_{m=1}^{n} mT^m p^{m-1} k_m} \\ &- \frac{\sum_{m=0}^{n} T^m p^{m+1} k_m \tau_m}{\sum_{m=1}^{n} mT^m p^{m-1} k_m}. \end{aligned} \tag{17}$$

Eq. (17) is the approximating equation for ϵ taking into consideration the effect of the frequency characteristics of the amplifiers.

Examples of Applications of the General Formula for ϵ

Circle test: Example 1

The computer setup in this case is shown in Fig. 4. The equation to be solved by it is

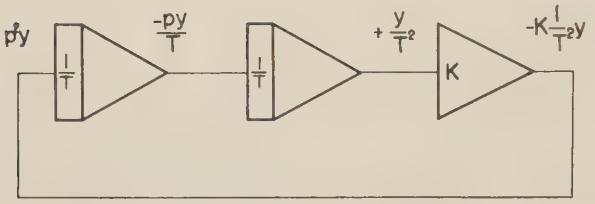


Fig. 4—Computer setup to solve equation (18).

TABLE II
CONSTANTS OF $Km(p)$ FOR USE WITH FIG. 4

m	km	Km
2	1	0
1	0	0
0	K	$1+K$

$$T^2p^2y + Ky = 0. \tag{18}$$

The result of finding Km by means of Table I is shown in Table II. Putting the result from Table II into (15), the error $C(p)$ of the coefficient setting elements is

$$C(p) = \frac{K}{\mu_K}\frac{1+K}{2T^2p} - \frac{K\tau_m}{2T^2} = \frac{K}{\mu_K}\frac{1+K}{2}\frac{p}{p^2T^2} - \frac{K}{2}\frac{\tau_m}{T^2}.$$

Substituting the value $p=j\sqrt{K}/T$ obtained from (18) into the above equation, we have

$$C(p) = -\frac{1+K}{2\mu_K}p + \frac{1}{2}p^2\tau_m$$

$$\therefore\ \epsilon = -\frac{1}{\mu_iT} - \frac{p}{\mu_i} - \Delta(p) - \frac{1+K}{2\mu_K}p + \frac{1}{2}p^2\tau_m. \tag{19}$$

The result of (19) is quite the same result that was obtained already.[5]

Tandem connection of N-integrators and M-constant coefficient multipliers: Example 2 (Fig. 5)

In this case the values of $Km(p)$ are given by Table III. Therefore, from (15)

$$C(p) = \frac{1}{\mu_K}\frac{\sum_{m=1}^{M}(1+C_m)\cdot\prod_{m=1}^{M}k_m}{NT^Np^{N-1}} - \frac{\sum_{m=1}^{M}\tau_m\cdot\prod_{m=1}^{M}k_m\cdot p}{NT^Np^{N-1}}. \tag{20}$$

In considering the relation

$$p^NT^N = -\prod_{m=1}^{M}k_m$$

$$C(p) = \frac{1}{N}\left\{-\frac{p}{\mu_K}\sum_{m=1}^{M}(1+C_m) + p^2\sum_{m=1}^{M}\tau_m\right\}. \tag{21}$$

We can easily understand from (21) that the errors of the coefficient setting elements in this case are proportional to the reciprocal of the number of integrators and increase as the algebraic sum of errors of each of the constant coefficient multipliers. This result is very useful in estimating the computing errors of each loop in case a complicated circuit setup is being treated.[10]

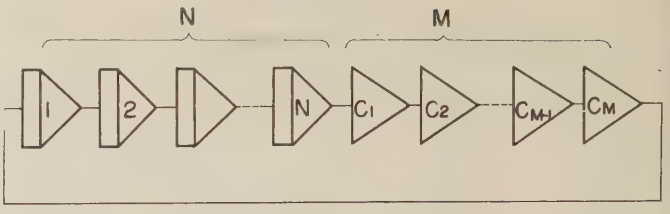


Fig. 5—Computer setup to solve the equation

$$p^NT^Ny \pm \prod_{m=1}^{M}K_my = 0.$$

TABLE III
CONSTANTS OF $Km(p)$ FOR USE WITH FIG. 5

m	km	Km
N	1	0
$N-1$	0	0
$N-2$	0	0
—	0	0
1	0	0
0	$\prod_{m=1}^{M}Km$	$\sum_{m=1}^{M}(1+Cm)$

Computing errors in the case of $0\leq k_m\leq 1$: Example 3

In this case we can use only sign changers and coefficient setting potentiometers as coefficient setting elements. This circuit has been analyzed by Marsocci (Fig. 6). In this case the values of $Km(p)$ are shown in Table IV; therefore in (15) we put $\tau_m\simeq 0$ and

$$C(p) = \frac{1}{\mu_K}\frac{2\sum_{m=0}^{n-1}\delta_mk_mp^m}{\sum_{m=1}^{n}mT^mp^{m-1}k_m}, \quad \text{where } \delta_{m=0} \text{ if } m \text{ is odd.}\ \ \delta_{m=1} \text{ if } m \text{ is even.} \tag{22}$$

Example 4

Let us consider the computer setup shown in Fig. 7. This circuit was built up so as to obtain the same result as in the case of the circle test without any kind of minor loop by giving positive damping by loop K' and negative damping by loop K''. If the characteristics of the sign changer are ideal, and the input characteristics of the two summing integrators are the same, we can examine the magnitude of the errors of the minor loop by comparing the setup having loops K', K'', with the one having no minor loop. We then calculate the increment of the theoretical value of the errors due to the minor loop. In this case the constants of $Km(p)$ are shown in Table V by virtue of Fig. 7. Substituting these values into (15) (in this case we may put we $\tau_m\simeq 0$), obtain

[10] We can separate all kinds of circuits into several separate integrators and coefficient setting amplifiers, no matter how complicated the original circuits may be.

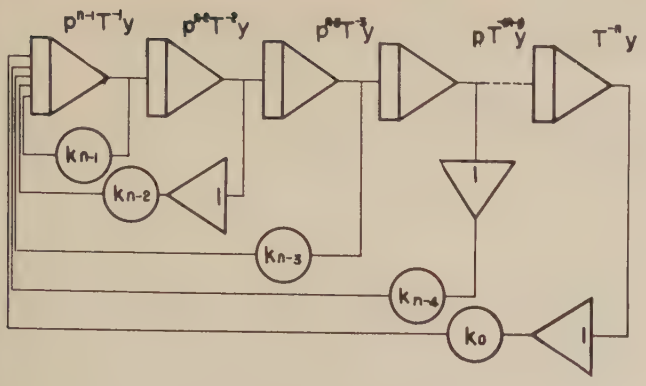


Fig. 6—Computer setup to solve the equation

$$\sum_{m=0}^{n} T^m p^m k_m y = 0$$

where $0 \leq km \leq 1$.

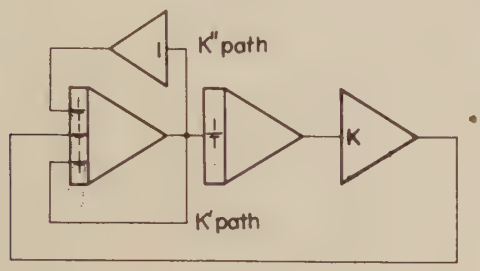


Fig. 7—Computer setup to solve the equation

$$p^2T^2y + Ky = 0.$$

TABLE IV

CONSTANTS OF $Km(p)$ FOR USE WITH FIG. 6

m	km	Km
n	1	0
$n-1$	k_{n-1}	0
$n-2$	k_{n-2}	2
$n-3$	k_{n-3}	0
$n-4$	k_{n-4}	2
—	—	—
2	k_2	2
1	k_2	2
1	k_1	0
0	k_0	2

$$C(p) = \frac{-2Tp + K(1+K)}{2T^2p}\frac{1}{\mu_K} = -\frac{1}{\mu_K T} + \frac{K(1+K)}{2\mu_K}p. \qquad (23)$$

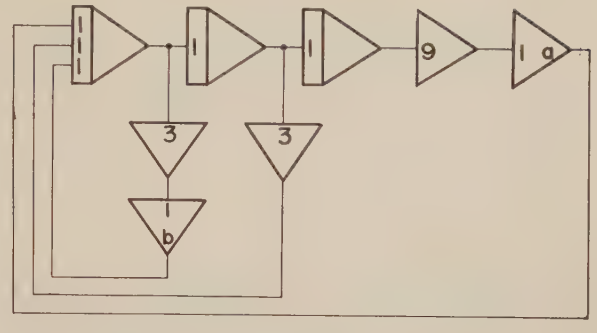
The first term on the right-hand side of (23) is due to the increment of the error from the minor loop.

Example 5

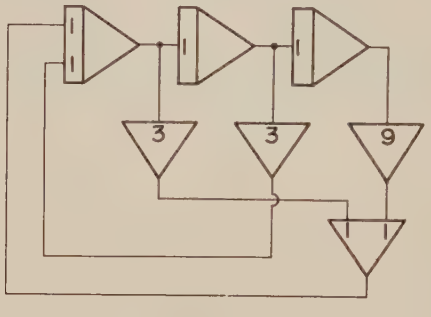
Next we consider a linear differential equation having a continuous oscillatory solution of order three.

$$p^3T^3y + 3p^2T^2y + 3pTy + 9y = 0 \quad p = \pm j\sqrt{3}/T. \qquad (24)$$

We may think of Fig. 8(a) as a computer setup for this



(a)



(b)

Fig. 8—(a) Computer setup to solve the equation

$$p^3T^3y + 3p^2T^2y + 3pTy + 9y = 0.$$

(b) Computer setup to solve the equation

$$p^3T^2y + 3p^2T^2y + 3pTy + 9y = 0.$$

TABLE V

CONSTANTS OF $Km(p)$ FOR USE WITH FIG. 7.

m	km	Km
2	1	0
1 K' path	1	0
1 K'' path	−1	2
0	K	$1+K$

equation. The table for $Km(p)$ is shown in Table VI referring to this figure.

Therefore from (15) and Table VI, we have

$$C(p) = \frac{1}{\mu}\frac{18p^2T^2 + 12pT + 108}{3p^2T^3 + 6pT^2 + 3T}.$$

Putting $p^2 = -3/T^2$ into above equation, we obtain

$$C(p) = -\frac{7}{4}\frac{p}{\mu} + \frac{15}{4}\frac{1}{\mu T}. \qquad (25)$$

Fig. 8(b) may be considered as a setup circuit for (24) and we can replace two sign changers a and b with a single adder. The constant, $Km(p)$, in this case may also be obtained from Table V referring to the standard of Table I.

CONCLUSIONS

It is clear from the above examples that the new general formula is closely connected with practical computer setups and that the errors are easily calculated when the circuit is given. Moreover, the errors which are calculated by the conventional formula are included

TABLE VI
CONSTANTS OF $Km(p)$ FOR USE WITH FIG. 8

m	km	Km
3	1	0
2	3	6
1	3	4
0	9	12

as a special case of the general formula which has been developed here. Several practical examples are shown as examples of the application of this formula. The authors believe that the general formula for the errors from the tandem connection of N integrators provides some estimate of the magnitude of the errors which arise in the case of complicated operational circuits.

APPENDIX I

EFFECTS OF THE DC AMPLIFIER GRID CURRENT AND DRIFT VOLTAGE ON ϵ

In the previous analysis of ϵ, we assumed that the grid current i_g and the drift voltage e_d of the dc amplifier may be neglected. Now if we consider the effect of i_g and e_d, the following relation can be obtained between y_m and Y_{m-1} of the output of the mth and the $(m-1)$th integrator from the right-hand side in Fig. 1.

$$y_{m-1} = I(p)(y + e_d - i_g R)_m - e_{dm}. \tag{26}$$

In (26), the subscript m of the bracket shows that the subscripts of the quantities in the bracket are all m. The computer setup shown in Fig. 1 is for the solution of the equation.

$$\sum_{m=0}^{n} k_m y_m = 0. \tag{27}$$

From (26) and (27) the following equation can be derived by putting $Y_n = y$

$$y_m = I(p)^{n-m} y + \sum_{i=m+1}^{n} I(p)^{i-m}(e_d - i_g R)_i - \sum_{i=m+1}^{n} I(p)^{i-(m+1)} e_{di}. \tag{28}$$

By substituting (28) into (27), one obtains

$$\sum_{m=0}^{n} k_m y_m = I(p)^n \sum_{m=0}^{n} k_m y I(p)^{-m} + \sum_{m=0}^{n} \sum_{i=m+1}^{n} k_m I(p)^{i-m}(e_d - i_g R)_i - \sum_{m=0}^{n} \sum_{i=m+1}^{n} k_m I(p)^{i-(m+1)} e_{di} = 0.$$

By dividing both sides of the above equations by $I(p)^n$, one obtains

$$\sum_{m=0}^{n} k_m y_m I(p)^{-m} + I(p)^{-n} \left\{ \sum_{i=0}^{n} (e_d - i_g R)_i \sum_{m=0}^{i-1} k_m I(p)^{i-m} - \sum_{i=0}^{n} (e_{di}) \sum_{m=0}^{i-1} k_m I(p)^{i-m-1} \right\} = 0. \tag{29}$$

Since $I(p)^{-1}$ is a differential operator, then by assuming that e_d and i_g are constant and independent of time, the second term has only $(e_d - i_g R)_n$ left and the third term vanishes. Eq. (29) becomes

$$\sum_{m=0}^{n} k_m I(p)^{-m} y = -(e_d - i_g R)_n = \text{const.} \tag{30}$$

For the ideal case $I(p)^{-1} = p$

$$\sum_{m=0}^{n} k_m p^m y = \text{const.}$$

Obviously the result of (30) shows that the grid current and the drift voltage have effects only on the initial condition voltage but not on ϵ.

APPENDIX II

CALCULATION OF $Km(p)$

When the tandem connection of two constant coefficient multipliers with multiplication constants C_1 and C_2 is used, the transfer function $Km(p)$ is

$$K_m(p) = \frac{\mu}{1+\mu+C_1} \frac{\mu}{1+\mu+C_2} \cdot C_1 C_2 \simeq C_1 C_2 \left(1 - \frac{1+C_1}{\mu}\right)\left(1 - \frac{1+C_2}{\mu}\right) \simeq C_1 C_2 \left\{1 - \frac{1}{\mu}[(1+C_1)+(1+C_2)]\right\} = k_m \left(1 - \frac{K_m}{\mu}\right), \qquad K_m = \sum_{m=1}^{2} (1 + C_m).$$

ACKNOWLEDGMENT

The authors wish to thank Z. Abe, chief laboratory member, who gave helpful advice, and J. Kawasaki, the investigator.

BIDEC—A Binary-to-Decimal or Decimal-to-Binary Converter*

JOHN F. COULEUR†

Summary—**Simple, high-speed devices to convert binary, binary coded octal, or Gray code numbers to binary coded decimal numbers or vice versa is described. Circuitry required is four shift register stages per decimal digit plus one 30-diode network per decimal digit. In simple form the conversion requires two operations per binary bit but is theoretically capable of working at one operation per bit.**

Introduction

THE following article is a description of devices to convert between the binary and decimal form of numbers. The method of conversion imposes no limitation on the number of digits and, in proposed advanced form, requires a time equal to the clock period times the number of digits to perform the conversion.

BIDEC is the result of a need for a fast, simple, binary-to-decimal converter to be used as the output buffer of a scale factor computer to reduce missile flight data. A study of conventional techniques indicated that time available for the conversion ruled out the use of counters. Complexity, size, and cost made the conventional division by 10 undesirable, and word length eliminated matrix conversion. Several other special methods of converting binary to decimal and decimal to binary were found,[1] but they either had inherent speed limitations or became increasingly complex with the number of digits to be converted.

The resulting search for an entirely new approach to the conversion of both binary to decimal and decimal to binary led to the formulation of the principles of operation of BIDEC, the original embodiments of which will be treated in this article. In the simplest form, either a binary or reflected binary number of any length in serial or parallel form may be converted to a binary coded decimal number, in a length of time equal to 2× number of binary digits × shifting time. Circuit requirements consist of four shift register stages per decimal digit, one, 30-diode logic per decimal digit, and a clock to count shift pulses. A change in wiring enables the conversion of decimal to binary, and an increase in complexity permits either conversion in the same device.

Theory

Making use of the fact that the decimal value of any digit in a binary number is 2^{N-1}, where N is location of the digit in the number, the decimal value of a binary number may be determined by adding appropriate powers of 2 as indicated by the presence of 1's in the binary number. The powers of 2 may be readily calculated by doubling one $N-1$ times or by multiplying the 1 in the binary number by two $N-1$ times.

A shift register is a convenient method of multiplication of a binary number by 2 since shifting the entire register once towards the most significant end doubles the value of all binary digits in the register if the stages are assigned arbitrary values of multiples of 2. With assignment of values to the stages of the shift register being somewhat arbitrary, there is no reason why the shift register may not be broken up into groups of four stages to be called decades. The four stages within the decades are assigned values of 1, 2, 4, and 8 ascending in the direction of shift, but the decades are assigned decimal values of 1, 10, 100, etc., ascending in the same direction. If a binary number of N digits is shifted into the register (most significant digit first), the most significant digit will be shifted and supposedly doubled $N-1$ times, and the following digits will also be shifted and doubled an equally appropriate number of times.

Obviously, the number in the register is not the decimal equivalent of the binary number and a quick examination will show why. The appropriate power of 2 was to be computed by doubling at each shift, but when a 1 was shifted from the "8" of any decade to the "1" of the next higher decade it only increased in value from 8 to 10. (Remember, the assignment of values was arbitrary.) Accordingly, a value of 6 was lost somewhere, and if the computation is to hold true, the 6 must be replaced. Therefore, 6 is added to the decade from which the one was shifted.

After the shifting is completed, only the necessity to clear out numbers over 10 in the decades by subtracting 10 and adding 1 to the next higher decade remains, and the conversion from binary to decimal has been accomplished.

While the above process performs the desired conversion, it is quite cumbersome to instrument and several simplifications may be made. Remembering that shifting to the left doubles the number in the register, 3 may be added to a decade when a "1" is sensed in the "8" stage since the subsequent shift will transfer the "1" to the next higher decade and double the 3 to the necessary 6.

To eliminate the possibility of a carry from the addition propagating into the next higher decade with the resulting necessity for adding 6 again, and to eliminate the clearing operation at the end of the conversion, it is desirable to prevent a decade from ever containing a number greater than 10. If the number in the register is 5, 6, or 7 prior to the shift, it will become a number

* Manuscript received by the PGEC, July 22, 1958. Revised manuscript received, August 7, 1958.

† General Electric Co., Syracuse, N. Y.

[1] Richards, Arithmetic Operations in Digital Computers, *et al.*

greater than 10 after the shift, and 5 should be subtracted from the decade and a 1 shifted into the next higher decade. In practice, this could consist of setting the "8" stage to "1" and subtracting 5, prior to the shift. This is the equivalent of adding 3, however, since $+8-5$ is equal to $+3$. A rule then develops: if the number in any decade is 8 or greater, add 3 before shifting, and if the number is 5, 6, or 7, add 3 prior to shifting. Or, in simpler terms, if the number in a decade is 5 or greater, add 3 prior to shifting. Since the number after shifting can never be greater than 9, only five possible cases exist to which 3 must be added: 5, 6, 7, 8, or 9. There is no carry to a higher decade, and each decade can stand as a separate entity.

It should be noted that there is no limit to the number of decades which may be cascaded. One decade is required for each decimal digit in the final answer.

The conversion from decimal to binary is based on repeated division of the decimal number by 2 with the remainders forming the binary number. With the decimal number entered in a shift register, divided in decades similar to that used for the binary to decimal converter, the division by 2 is performed by shifting toward the least significant end of the register.

When a 1 is shifted from the "1" stage of decade to the "8" stage of the next lower decade, however, a division by two is not performed since the value of the 1 changes from 10 to 8 rather than 10 to 5. Obviously, a value of 3 has been gained, which must be subtracted. If, after a shift, a decade has a 1 in the "8" stage, it can only have come from the next higher decade and, accordingly, the rule may be formulated: if, after the shift pulse, the number in a decade is 8 or greater, 3 must be subtracted. For obvious reasons there are again five possible numbers from which 3 must be subtracted: 8, 9, 10, 11, and 12. There is no borrow to the next decade, and again each decade can stand as a separate entity. The number of shift pulses to be applied is equal to the number of binary digits in the largest number which can be contained in the register.

Binary-to-Decimal Converter

The process of converting a number in binary form to its decimal equivalent consists of shifting the number into a shift register having $4N$ stages grouped to form N decades, the contents of each decade forming a decimal digit in binary coded form, and the number of N decades required being determined by the decimal equivalent of the binary number. The conversion process consists of shifting the binary number into the register one bit at a time, most significant digit first, testing the content of each decade prior to each shift and adding binary 3 to any decade which contains binary 5 or greater.

Fig. 1 is a block diagram of the converter, Fig. 2 is a block diagram of the logic circuitry associated with each of the decades, and Fig. 3 is an illustration of the operation of the converter.

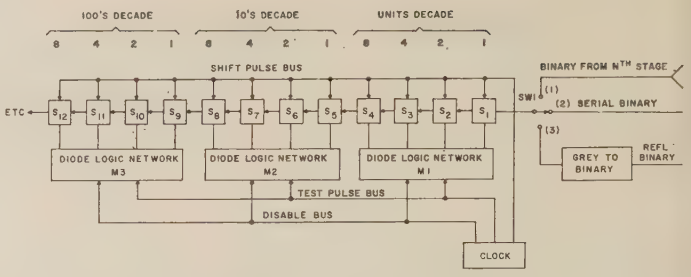


Fig. 1—Block diagram, binary-to-decimal converter.

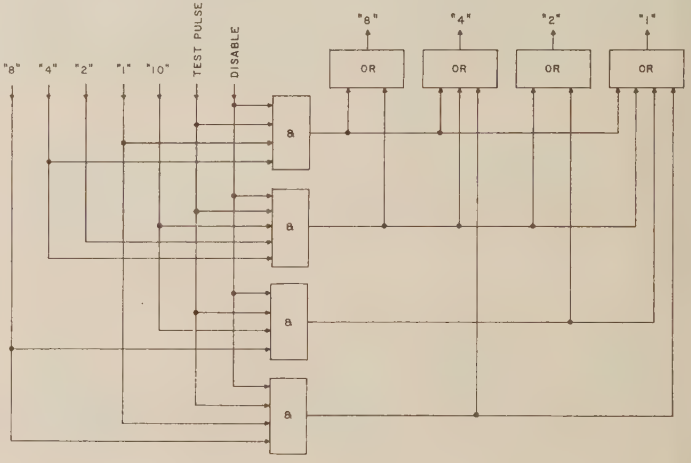


Fig. 2—Binary-to-decimal logic network.

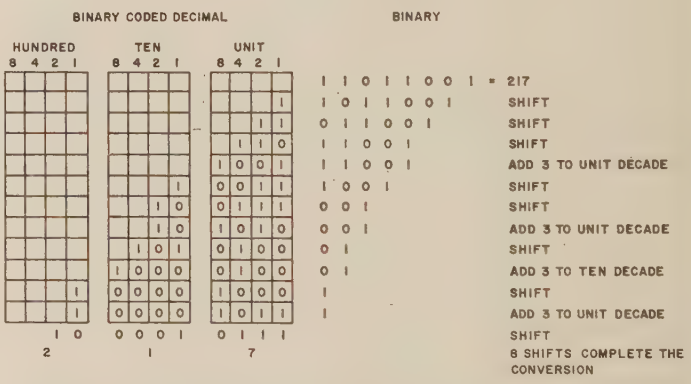


Fig. 3—Example binary to decimal. Note: prior to each shift, each decade is tested for 5 or greater, and, if so, 3 is added.

Referring to Fig. 1, there is shown a shift register which, for example, consists of 12 stages, S1 through S-12. The four stages S1–S4 plus network M1 form the units decade of the shift register, stages S5–S8 plus network M2 form the 10's decade, and stages S9–S12 plus network M3 form the 100's decade. Each stage of a decade, of course, will contain a binary "1" or a "0" and the stages within the decades have binary weights of 8, 4, 2, and 1 increasing in significance in the same direction as the decimal weights. For convenience, the stages within a decade will be referred to by their binary weight; *i.e.*, the "4" stage would refer to stages S3, S7, or S11. The shift register is connected to a clock, or source of shift pulses, in such a manner that each shift pulse shifts the number in the register one stage to the left, or toward a more significant value.

A second output of the clock preceding each shift pulse, is the test pulse applied to the diode networks shown in Fig. 2. In essence, the networks determine if the number in a decade is 5 or greater and, if so, add 3 by complementing the appropriate stages of the decade. For example:

0101 (5) should become 1000 (8),
0110 (6) should become 1001 (9),
0111 (7) should become 1010 (10),
1000 (8) should become 1011 (11), and
1001 (9) should become 1100 (12).

Examination will show:

If "4" and "1" are 1's, "8," "4," and "1" stages are complemented.

If "4" and "2" are 1's and "1" is 0, "8," "4," "2," and "1" are complemented.

If "8" is 1 and "1" is 0, "2" and "1" are complemented.

If "8" and "1" are 1, "4" and "1" are complemented.

The network contains the necessary logic to generate the proper pulses to be applied to the complementing inputs of the shift register stages. Many other networks, or adders, are available where the shift register stage does not lend itself to being complemented.

The clock contains a counter, and after a number of test and shift pulses equal to the number of binary digits in the word has been applied to the converter, the process is complete and the decimal number is in the converter in binary coded form.

The description of the conversion has been for a binary number in serial form. Switch 1 permits the conversion of a binary number in parallel form or a reflected binary number in parallel or serial form.

For the conversion of a binary number in parallel form, the binary number is read in parallel into the shift register with the most significant digit in the most significant stage (S12 in the example) and SW1 is put in position 1 connecting stage S1 to S12. Application of shift pulses causes the conversion to be accomplished as previously described. To prevent digits of the binary number from being converted prematurely, the networks are prevented from generating complementing pulses until the parallel-entered binary number has been shifted out of the decade. This is accomplished by means of the "disable" bus shown on the network in Fig. 2. The clock generates disable gates during the counting period in such a manner that the units decade network is "enabled" after three shift pulses, the 10's decade after six shift pulses, the 100's decade after nine shift pulses, etc., the number three deriving from the fact that three shifts are necessary to enter a number greater than 5 into a decade.

To convert either serial or parallel Gray code to decimal, the same converter is used with the addition of a serial Gray-to-binary converter at the input to stage S1. The Gray code converter consists of an "and" of the input stage and the test pulse. If the input stage is a

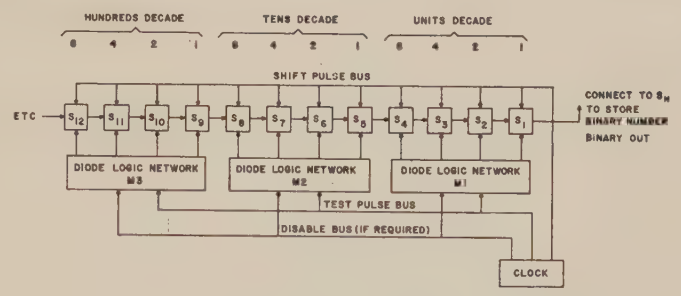


Fig. 4—Block diagram, decimal-to-binary converter. Note: shift register shifts to right.

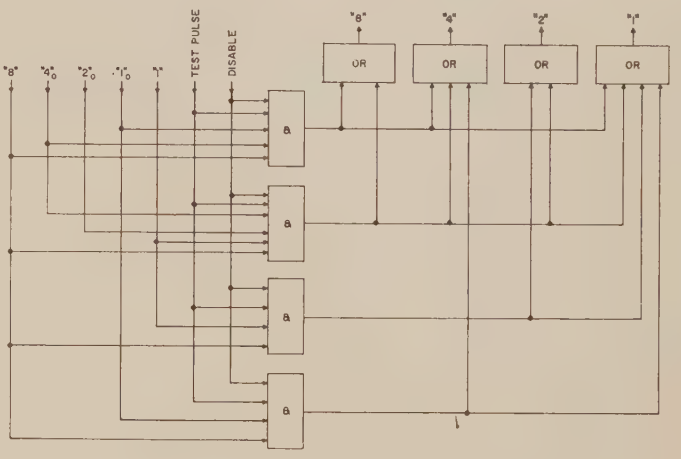


Fig. 5—Decimal-to-binary logic network.

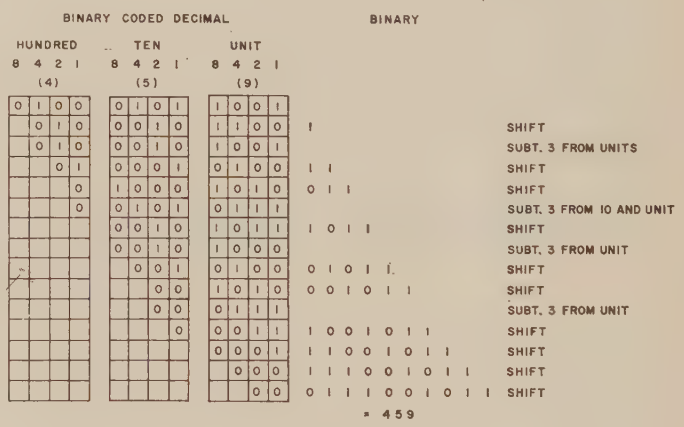


Fig. 6—Example decimal to binary. Note: prior to each shift, each decade is tested for 8 or greater, and, if so, 3 is subtracted.

"1," a pulse is generated which complements either a pre-input flip-flop, if the input is serial, or the last stage, if the parallel storage method is used.

Decimal-to-Binary Converter

The process of converting a number in decimal (binary coded decimal) form to its binary equivalent consists of shifting the number out of a shift register having $4N$ stages grouped to form N decades, the contents of each decade forming a decimal digit. The conversion process consists of shifting the binary number, one bit at a time, least significant digit first, out of the register, testing the content of each decade after the shift and

subtracting three from any decade containing eight or greater.

Fig. 4 is a block diagram of the converter, Fig. 5 is a block diagram of logic circuitry associated with each of the decades, and Fig. 6 is an illustration of the operation of the converter.

Referring to Fig. 4, there is a shift register and clock arranged in a manner similar to that of Fig. 1 except that the shift is toward the least significant digit, and the decimal number in binary coded form has been entered in the register in parallel form. A second output of the clock, following the shift pulse, is the test pulse, which is applied to the diode networks shown in Fig. 5.

The networks determine if the number in the decade is 8 or greater and, if so, subtract 3 by complementing the appropriate stages of the decade. For example:

1000 (8) should become 0101 (5).
1001 (9) should become 0110 (6).
1010 (10) should become 0111 (7).
1011 (11) should become 1000 (8).
1100 (12) should become 1001 (9).

Examination will show that:

If "8" is 1 and "4" and "1" are 0, "8," "4," and "1" are complemented.
If "8" is 1, "1" is 1, and "4" and "2" are 0, "8," "4," "2," and "1" are complemented.
If "8" is 1 and "1" is 1, "2" and "1" are complemented.
If "8" is 1 and "1" is 0, "4" and "1" are complemented.

The network contains the necessary logic to generate the proper complementing pulses.

After a number of shift and test pulses equal to the number of bits of the largest binary number which could be represented by a decimal number of N digits the conversion is complete, and the decimal number has been shifted out of the converter in serial binary form, least significant digit first.

If the decimal number is in serial form, it may be shifted into the register prior to the conversion. If the binary number is desired in parallel form, switch SW-1 is thrown to position 2 and the binary number circulated back into the shift register. However, disable gates must be generated to prevent complementing the binary number.

Conclusion

The circuit design of either of the converters is straight forward and there are many techniques available to provide the shifting and testing functions. The model built used flip-flop shift register stages and dc logic. As the stages complemented in the testing operation affect the logic, it is necessary to delay the stage inputs to the logic by approximately the width of the test pulse.

Further study indicated that theoretically, the speed of conversion could be doubled at the expense of some complexity by eliminating the "test" operation. Essentially, the presence of a 5 or greater in a decade would block the shift pulse to the "8," "4," and "2" stages of that decade, and these stages would be set to the number plus 3 as it would have been shifted. Using 1-mc stages, a 20-digit binary number or 6-digit decimal number could thus be converted in 20 μsec.

Diodeless Magnetic Shift Registers Utilizing Transfluxors*

NOAH S. PRYWES†

Summary—**Shift registers using magnetic cores have been conventionally designed with diodes to provide isolation between stages. It is possible to use transfluxors in a two-core-per-bit fashion so as to provide isolation between stages by magnetic means rather than with diodes. A design procedure for doing so is discussed.**

I. Introduction

MAGNETIC materials with a characteristic as shown in Fig. 1 can be used for information storage. Their main useful features are two opposite remanent magnetizations at saturation and a minimal coercivity H_c required for a domain change in a process of magnetization reversal.

To explain further the magnetic characteristic, consider, for instance, an initial situation of $+B_s$ and $H=0$ in Fig. 1. For a positive or a negative field intensity smaller than H_c, the nearly horizontal line at the top of the characteristic in Fig. 1 will represent the corresponding changes. At the removal of such a field, return to the magnetization of $+B_s$ and $H=0$ will occur. Thus any change of flux at the application of such a field was reversible. However, if a negative field in excess of H_c is applied, an irreversible flux change occurs along the nearly vertical part of the left of the major hysteresis loop shown in Fig. 1. The average rate of change of such flux along this line is approximately proportional to the amount by which the applied field exceeds the coercive field H_c. If the applied field is removed before the complete reversing of magnetization along the major hysteresis loop, the return to $H=0$ will occur along a

* Manuscript received by the PGEC, September 3, 1958.
† Moore School of Elec. Eng., Univ. of Pennsylvania. Formerly with Remington-Rand UNIVAC, Philadelphia, Pa.

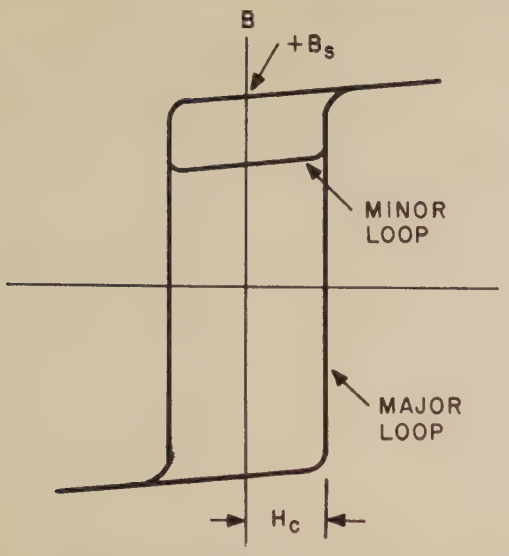


Fig. 1—Magnetic characteristic. B = flux density, H = field intensity, H_c = coercive field.

minor hysteresis loop corresponding to the average remanent flux density at the instant of removal of the applied field. This description of the magnetic characteristic should be sufficient for the purpose of this discussion. Other secondary phenomena will be assumed to have a negligible effect.

As bistable devices, magnetic cores may be assembled to constitute a shift register. It has been found that certain isolation must be provided between stages; this so far has been available through the use of diodes. [1] The invention of the transfluxor [2],[3] reveals a method by which the desired isolation can be achieved without diodes. This paper will discuss some registers in which such advantage is taken. Fundamentally they are of the "two-core-per-bit" kind, [1],[4] with the isolation between stages provided by magnetic means rather than diodes. A design procedure for a shift register utilizing cores and windings only will be discussed. Production costs for such a register would be low with printed-circuit techniques.

II. A Three-Aperture Transfluxor

A short description of the operation of the transfluxor will be given here, but a complete explanation and description is given by Rajchman [2], [3].

The transfluxor shown in Fig. 2 can be used for a stage of the proposed shift register. It is a core which consists of four legs with a small aperture between legs 1 and 2, a small aperture between legs 3 and 4, and a large aperture between legs 2 and 3. It is made of ferrite material which has the loop characteristic shown in Fig. 1. In addition to its characteristic described in the Introduction, the requirement of continuity of flux lines (div $B = 0$) will determine any flux changes. In order to change the flux around the big center hole, the ampere turns of the winding must produce a magnetic field exceeding H_c along that long path. Fewer ampere turns are needed to change the flux along the much shorter paths around the small holes. Thus it is possible to

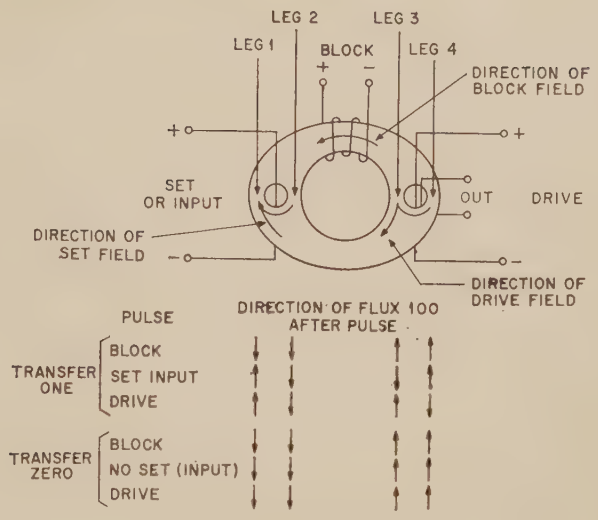


Fig. 2—Three-aperture transfluxor.

change the flux around one small hole without affecting the windings through the other small hole, provided enough ampere turns are applied to exceed H_c in the short path around the small hole, but fewer than are required to reach H_c along the longer path around the large hole. The required isolation is achieved, in that information can be read out by flux change around the right-hand small hole without affecting the input winding through the left-hand small hole. When a shift of flux is required, a larger pulse applied to the input winding through the left-hand small hole reads information into the area around the right-hand small hole.

The transfer of a "one" and of a "zero" is shown at the bottom of Fig. 2. The arrow below each of the legs of the transfluxor indicates the direction of remanent flux at saturation in that leg. The arrows show the direction of the remanent fluxes after application of the pulse named.

Transfer of a One

The first line shows the remanent fluxes after a block pulse. This situation also corresponds to a zero state. The flux polarities are downward in legs 1 and 2 and upward in legs 3 and 4. The condition of continuity of flux in the ferrite is thus satisfied.

A set pulse, representing a one, causes a clockwise field around the left-hand small hole. This field is of sufficient amplitude to exceed H_c in the path round legs 1 and 3. The flux in leg 2 can change only slightly since it is already in saturation in the downward direction. The change in flux in leg 1 must then be compensated by an equal and opposite flux change in legs 3 and 4. The path of flux change through leg 3 is shorter than that through leg 4, and thus will be preferred. Also, leg 4 can be biased to prevent a change of flux in it. A flux change in leg 3 then necessarily compensates the change of flux in leg 1 to satisfy the continuity of flux condition.

A drive pulse, applied after the set pulse, changes the flux around the small hole on the right, producing an output pulse. The input winding, however, is not af-

fected because of the limitations on the amplitude of the drive ampere turns.

Transfer of a Zero

The absence of a set pulse represents a zero. In the absence of a set pulse, the core remains in the block state. When a drive pulse is applied, the field caused cannot affect the flux in legs 1 and 2 because it does not exceed the value of H_c along the path around the large center hole. It cannot change the flux in leg 3 as this is already at saturation in the direction of the applied field. No significant flux change in legs 1, 2, and 3 is possible. Hence, no flux change can occur in leg 4 because any change would have to be compensated by equal and opposite changes in other legs to satisfy the condition of flux continuity. Thus there is no significant change in flux linking the output winding, resulting in lack of input to the following core, or a zero output. The process of transfer and discrimination between a zero and a one has thus been demonstrated.

III. A Diodeless Two-Core-Per-Bit Shift Register

Before analyzing the operation of a transfluxor stage, it would be well to present a general description of how these cores may be connected to form a shift register. Figs. 3 and 4 illustrate the operation of a two-core-per-bit shift register. The register shown at the top of Fig. 3 is capable of holding four bits. The arrows show the pattern of remanent fluxes in all stages for the process of shifting "1100" into the register. Fig. 4 diagrams the timing of supply pulses and some input pulses.

The cores in Fig. 3 are shown with rectangular apertures only for convenience of presentation. Legs 1, 2, 3, and 4, input, output, drive, cancellation, and block windings are labeled on the first core of the left of Fig. 3. The windings on the other cores are similar. The output winding of each core is connected to the input winding of the successive core. The winding sense of all windings and the series grouping of windings, so that they may be fed by a single current pulse, are also shown.

The windings differ somewhat from those in Fig. 2. The input goes to leg 2 instead of leg 1 in order to shorten the path of flux changes during a one input. When the input winding is on leg 2, an input pulse reverses the flux in a path surrounding only the large center hole. On the other hand, when the input winding is on leg 1, as in Fig. 2, the input pulse changes the flux around the longer path which surrounds both the large center hole and the small hole at the left. Placing the input winding on leg 2, therefore, requires fewer ampere turns in the input. The drive winding is wound on leg 3 rather than leg 4 in order to lessen air flux linkages between drive and output windings. A cancellation winding, wound in opposition to the block winding, has been added. This cancellation winding raises the upper limit on the drive-pulse ampere turns.

The lower part of Fig. 3 diagrams the process of shifting a pattern of information. Again the arrows under each leg represent the direction of magnetization at saturation. With the timing shown in Fig. 4 a source of four current pulses is necessary. These pulses are Drive I, Block I, Drive II, and Block II. Since this is a two-core-per-bit register, Drive I and Drive II pulses are applied to alternate cores. The block pulses follow the drive pulses and reset the cores to the zero state. Block pulses are required because the transfluxor, by contrast with the single-aperture core, has nondestructive readout. This seeming disadvantage can be turned to advantage if an additional small hole similar to the other two small holes is provided, as shown in Fig. 5. Then reading out without shifting the information is possible.

A pattern of 1100 is shifted into the register to illustrate its operation as shown in the lower part of Fig. 3. The one input into the first core is in the form of a positive pulse at the time of Drive II, followed by a negative pulse at the time of Block II. (See Fig. 4.) The output from a stage into the successive stage follows the same pattern. Thus, if the output of the eighth core is connected to the input of the first core, the information is propagated around a ring. The flux patterns of cores in a one stage are framed to assist in examination of the shifting flux pattern which represents the information introduced.

The shift register may be modified to accept an input of all positive pulses, as shown by the alternative input wound on leg 1 in Fig. 3.

At Drive II the one enters the first core and flux changes in legs 2 and 3. The Block II pulse, which follows, reverses the direction of flux in legs 1 and 2. At Drive I the one enters the second core where flux changes in legs 2 and 3, while flux changes on legs 3 and 4 of the first core. Block I reverses the flux in legs 1 and 4 of the first core and legs 1 and 2 of the second core. On the next Drive II pulse the second one in the 1100 pattern enters the first core of the register, and the first one in the 1100 pattern is shifted to the third core.

With this concept of how information is shifted, a closer look can be taken at a single step in this process.

IV. Analysis of Information Transfer

Analysis of Flux Switching during Drive Pulse

In the three cores shown in Fig. 6, information is transferred from left to right. This analysis deals with a cycle in which information is transferred from the middle core. Supply windings are shown on the middle core only. The block winding with N_B turns provides a magnetomotive force (mmf) of B in the clockwise direction around the large aperture. The drive winding on leg 4, with N_D turns, provides an mmf of Δ in the clockwise direction around the small aperture at the right of the transfluxor core. An additional cancellation winding with N_C turns provides an mmf K in the counterclockwise direction. The K mmf is provided to cancel the effect of the Δ mmf around the large aper-

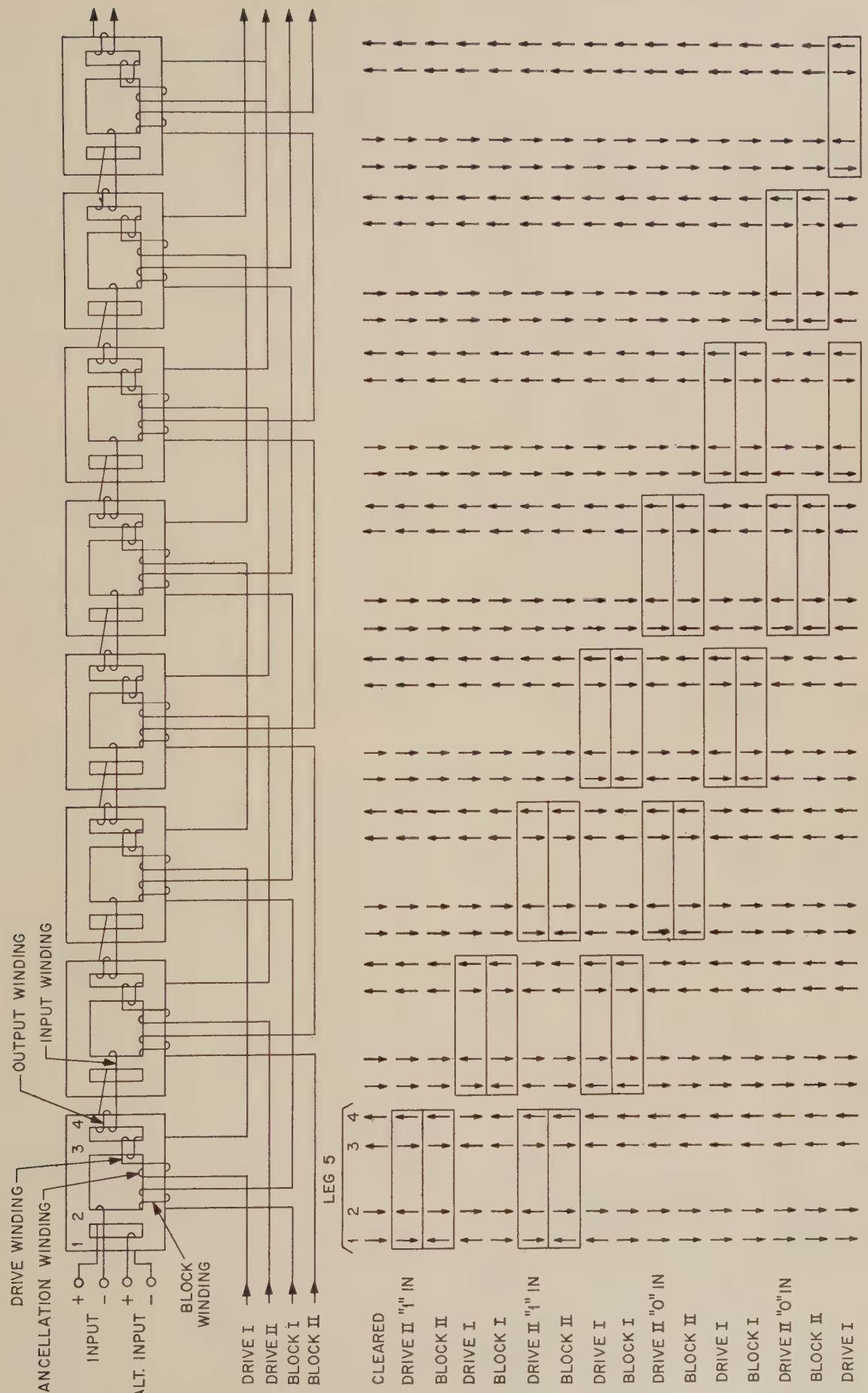


Fig. 3—Four stages of a diodeless shift register and flux pattern when 1100 is shifted through the register.

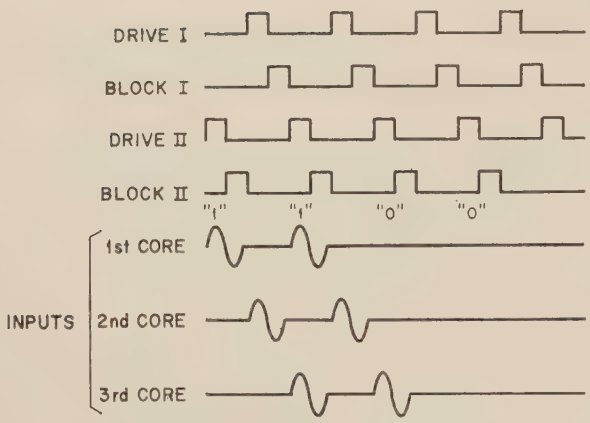


Fig. 4—Supply and input pulse pattern for the register of Fig. 3.

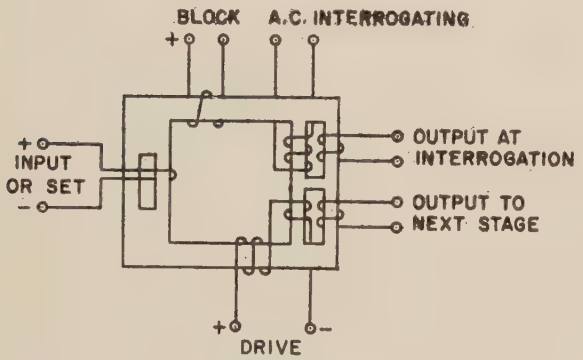


Fig. 5—Four-aperture transfluxor with nondestructive interrogation.

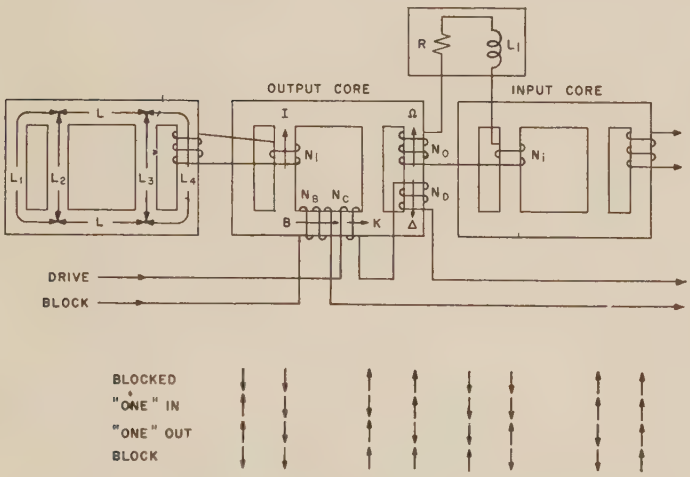


Fig. 6—Mmf's and their paths in information transfer.

ture; that is, to confine the effect of Δ to legs 3 and 4 only. Hence K must be coincident with Δ and is supplied by arranging the cancellation winding in series with the drive winding. The mmf's Ω and I are provided by the output winding on leg 4 with N_o turns and the input winding on leg 2 of the input core at right with N_i turns, respectively. The box connected in series with the output winding and the input winding of the core at left represents the resistance R and the self-inductance L_i of the input and output windings.

The process of reading a one from the middle core to the core at the right is illustrated in Fig. 6. The arrows in the diagram again represent direction of magnetization at saturation. The line marked "one in" shows the state of the core prior to the drive pulse. The line marked "one out" shows the flux pattern of the core after the drive pulse. The line marked "block" shows the flux configuration after application of the block pulse. In the case of a zero transfer both cores will remain in the block state.

To simplify the analysis it is assumed that the wall thickness is small enough to justify considering a uniform magnetic field in the core walls. Thus the magnetic paths and their lengths are depicted by lines L_1, L_2, L_3, L_4, and L on the left-hand core in Fig. 6. The subscripts refer to the respective legs.

When the drive pulse is applied, it is desirable that the flux be switched in legs 3 and 4 of the transfluxor core in the case of a readout of a one and that no appreciable flux change occur in the case of a zero. These requirements can be set into formulas with the aid of Fig. 6.

As seen in the flux diagram of Fig. 6, the following condition is imposed on operation during the drive pulse. In reading a one out of the center core, the path enclosing the large aperture and the small left aperture should not be affected. Hence the fields in this path should be smaller than H_c. That is,

$$K - \int_{L_3} H dL \leq H_c(2L + L_1). \tag{1}$$

When its direction is upward, H is taken as positive. Equal magnetic fields are required for equal rate of change in induction at any point in the core. Since the flux is being switched at an equal rate along the path L_3 and L_4, the following can be stated:

$$\int_{L_3} H dL = H_3 L_3;$$

$$\int_{L_4} H dL = H_4 L_4, \ \& \ H_3 = H_4 > H_c \tag{2}$$

and from (1):

$$K \leq H_c(2L + L_1) + H_3 L_3. \tag{3}$$

Also, the mmf's in the path in which there is a flux change (legs 3 and 4) follow the relationship:

$$H_3 L_3 + H_4 L_4 = H_3(L_3 + L_4) = \Delta - \Omega_{\text{one}}. \tag{4}$$

Let $M_{c1,3}$ be the mmf existing in the path of legs 1 and 3 when the field in this path is H_c:

$$M_{c1,3} = H_c(L_1 + L_3 + 2L).$$

Eqs. (3) and (4) can be written in the form:

$$K_{\max} = M_{c1,3} + L_3(H_3 - H_c) \tag{5}$$

$$\Delta_{\max} - \Omega_{\text{one}} = H_3(L_3 + L_4). \tag{6}$$

Both Ω_{one} and H_3 depend on load conditions and the ed of operation. Consider the case of the output con- cted in shunt to n input cores as in Fig. 7(a). The rrent in input and output windings is the same. The e of change of flux and the fields in legs 2 and 3 of the ut cores are equal. Since the cross section of mag- tic material in the sections denoted by lengths L can made much greater than that in legs 2 and 3, the plied field can be essentially H_c. Thus, we can write expression for Ω_{one}:

$$\Omega_{one} = n \frac{N_o}{N_i} \quad I = n \frac{N_o}{N_i} [H_2(L_2 + L_3) + 2H_cL] \quad (7)$$

ere the value of H_2 relates to leg 2 of the input core. t $M_{c2,3}=H_c(L_2+L_3+2L)$. Then (7) can be written the form:

$$\Omega_{one} = n \frac{N_o}{N_i} [M_{c2,3} + (H_2 - H_c)(L_2 + L_3)]. \quad (8)$$

In the case of a zero transfer, the drive pulse does not ect the cores. They remain in the block state. Again, s requires that the field in the path around the large erture be less than H_c. That is,

$$\Delta - \Omega_{zero} - K \leq H_c(L_4 + 2L + L_2)$$

$$\Delta - \Omega_{zero} - K > 0 \text{ and } \Omega_{zero} \ll \Omega_{one}. \quad (9)$$

ain, for $M_{c2,4}=H_c(L_2+L_4+2L)$ and using the maxi- um values of Δ, K, and Ω_{zero},

$$\Delta_{max} - K_{max} - \Omega_{zero} = M_{c2,4}. \quad (10)$$

In the case of a zero transfer, the real requirement is e absence of flux change in the input core at the right Fig. 6. Even though the flux in leg 4 of the middle re changes slightly, producing a small pulse in the tput winding, this does not amount to an information nsfer unless the flux also changes in leg 2 of the input re. Any small flux change in leg 4 of the output core erased by the following block pulse without affecting e operation. To cause a flux change in the input core e mmf Ω_{zero} must provide a field in excess of H_c along e path consisting of L_2, L, L_3, and L. The significance this observation is that in the case of a zero the flux ay change in leg 4 of the output core. Such a flux ange would be compensated by the voltage drop in e resistance R and self-inductance L_i, without causing irreversible flux change in the input core. The maxi- um mmf Ω_{zero} allowed in the case of a transfer of a zero ould then be:

$$\Omega_{zero(max)} = n \frac{N_o}{N_i} M_{c2,3} \quad (11)$$

here n is the number of inputs of cores connected in unt to the output of the driven core.

The rate of flux change in leg 3 of the output core is oportional to the difference between H_3 and H_c; that leg 2 of the input core is proportional to the difference between H_2 and H_c. Hence, the relationship between H_3 and H_2 is

$$N_o(H_3 - H_c) = N_i(H_2 - H_c). \quad (12)$$

It is desired to find the upper bound on Δ. An examination of the conditions leading to (11) shows that Δ is maximum when K also is maximum. Hence, in design the number of turns on the cancellation winding should always be such as to provide K_{max}. There are five equations [(5), (6), (8), (10), and (11)] with the five unknowns: Δ_{max}, K_{max}, H_3, Ω_{zero} and Ω_{one}. They can be solved for Δ_{max} and K_{max} which are required as design information. Since Δ_{max} and K_{max} vary somewhat for differently shaped cores, it is of little interest to write these expressions. However, certian approximations can be instructive. When legs 1, 2, 3, and 4 are approximately equal, and where L is large and $n=1$, Δ_{max} can be approximated to be

$$\Delta_{max} = 3M_{c2,3}. \quad (13)$$

The upper limit of drive mmf (Δ) imposed by (13) limits both the number of transfluxors (n) that a single transfluxor can drive and the speed of switching. When L, L_1, L_2, L_3, and L_4 are small and switching is extremely slow, n can be calculated. A minimum of one transfluxor and a maximum of n would operate satisfactorily. When switching is very slow and $n_o/N_i \approx 1$ then,

$$H_3 = H_2 = H_c$$
$$\Omega_{zero} = M_{c2,3}$$
$$\Omega_{one} = nM_{c2,3}.$$

This shows the solution for n_{max} from (5) through (12),

$$n_{max} = 3. \quad (14)$$

To determine the switching time of a transfluxor driving only one other transfluxor, as in a shift register, assume

$$L = L_1 = L_2 = L_3 = L_4$$
$$\frac{N_o}{N_i} \simeq 1$$

then,

$$H_2 = H_3.$$

Introducing these assumptions into (5) through (12):

$$H_3 = H_2 = 3H_c. \quad (15)$$

In an RCA transfluxor, a switching field of three times the coercive field provides 0.3-μsec switching time. The time can be improved by making L much greater than L_1, L_2, L_3, and L_4. A shift register using RCA transfluxors has been operated above 500 kc with a switching time of 0.8 μsec.

Turns Ratio of Output to Input Windings

To determine the ratio of input to output winding turns, the process of reading out a one will be analyzed.

Fig. 7(a) shows the transfluxor parts involved. Only legs 3 and 4 of the output core and legs 2 and 3 of the input core are shown since there is no flux change in the other legs. This simple situation, the transfer of information in a single-aperture core, has been treated previously by A. Wang [1]. Fig. 7(a) shows the flux configurations before and after a drive pulse.

In Fig. 7(c) the magnetic characteristic is represented by an equivalent circuit. The requirement that field H_c must be exceeded before flux change occurs can be represented as a current drain of M_c/N_o. The amount by which the applied field exceeds this value is used to accelerate the flux change. This excess can be represented by an equivalent resistance which is proportional to the square of the number of turns [5]. In the figure, the current drain of $(M_{c3,4})/N_o$ represents the coercive current in the output core, and $n(M_{c2,3})/N_i$ represents the coercive current in n input cores. Also, the applied drive field reverses the flux in input and output cores at the rate determined by the equivalent resistance. This resistance is R_0 times the square of the number of turns.

The ratio, N_o/N_i, should be greater than one in order to compensate for flux lost in the transfer. Also, N_o/N_i has an upper limit, a result of the requirement that drive field must exceed coercive drain.

$$\frac{\Delta}{N_o} \geq \frac{M_{c3,4}}{N_o} + n\frac{M_{c2,3}}{N_i}. \tag{16}$$

This, together with the lower limit, forms:

$$1 < \frac{N_o}{N_i} < \frac{\Delta - M_{c3,4}}{nM_{c2,3}}. \tag{17}$$

The rate of flux reversal in the input cores depends on energy supplied to the resistance $(N_i^2R_o)/n$. See Fig. 7(c). This energy is maximized if N_o/N_i is near enough to unity to compensate for flux loss. In practice, such refinement may require so many turns that it rules out printed circuit techniques. In the shift register that was operated, the number of turns used was $N_0=3$ and $N_i=2$.

The Blocking of the Buildup of Zeros

Fig. 7(a) illustrates the reversible and irreversible flux changes. The drive winding is magnetically linked to the output winding through the magnetic core, which is assumed to have only irreversible flux changes. Separate linear linkage m accounts for reversible flux changes. During the transfer of a zero, voltage V_o is induced through the linear linkage m. The amplitude of voltage V_o is

$$V_o = m\frac{dI_d}{dt} \tag{18}$$

where I_d is the drive current. If voltage V_o causes an mmf in the input winding in excess of $M_{c2,3}$, a small change of flux may occur in the path of legs 2 and 3 of the input core. Successive drive pulses amplifying this small change may eventually introduce a one. There are several methods for avoiding this undesirable process.

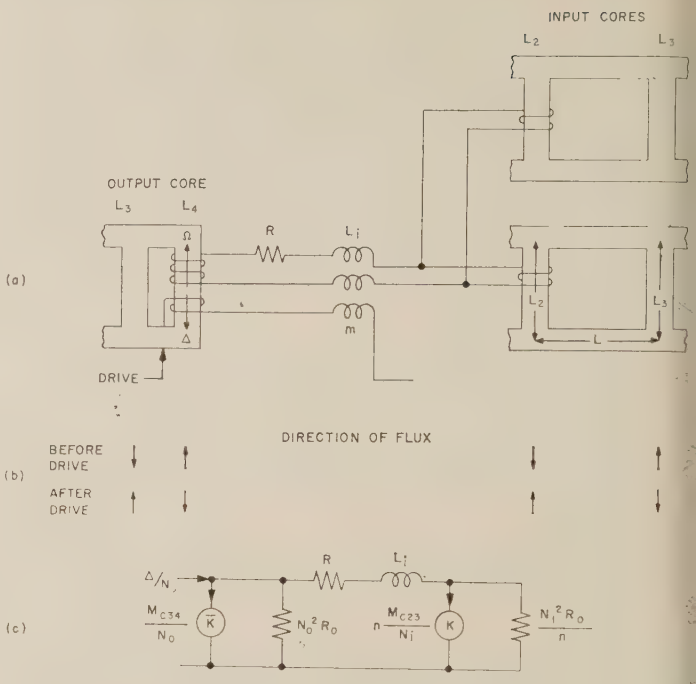


Fig. 7—Transfer of a one.

Resistance in the output circuit: A simple way of blocking zeros is to introduce a resistance, R, in the loops of the input and output windings such that

$$M_{c2,3} > \frac{V_o}{R}N_i. \tag{19}$$

The disadvantages of this method are 1) it complicates production, thus raising costs and 2) the resistors dissipate energy thus reducing speed.

Use of a buck core: The use of a buck core is known as well [4],[6]. A corresponding circuit configuration is shown in Fig. 8. The buck core in this circuit is driven at the application of each drive pulse inducing in each output a voltage in opposition to that induced by the transfer of a zero. During transfer of a one the blocking pulse is small and of short duration so as to subtract little flux from the output. Only the output circuits are shown in Fig. 8, with the rest of the circuit remaining the same as in Fig. 6.

Cancelling zero output during block pulses: Flux change due to zero output can be cancelled during the block pulse. This is done by reversing the flux in the path of leg 2 and 3 of the input core by an amount equal and opposite to the change caused by a zero output. The block pulse must then induce the corresponding voltage in the output circuit. One means is to provide sufficient linkage between the block pulse circuit and the output circuit and to control the rise times of the drive and block pulses. This approach has been found very simple and satisfactory in operating the shift register.

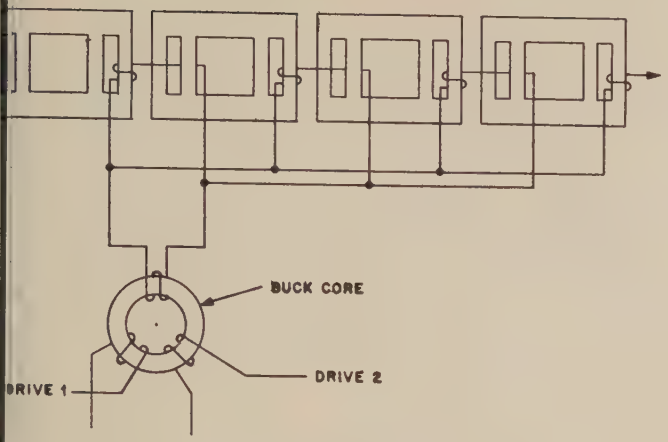


Fig. 8—Suppressing zero output by the use of a buck core.

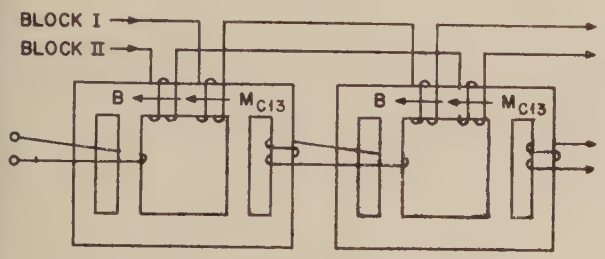


Fig. 9—Another arrangement to cancel zero buildup.

An arrangement where the linkage between the out-
t circuit and the block pulse circuit is provided in the
ut core is shown in Fig. 9.

Reducing linear flux linkage: By winding the drive
d output windings on legs 3 and 4, respectively, in-
ad of both on leg 4, the linear flux linkage coefficient
is decreased greatly. The two ways of winding with
e correct number of turns are shown in Fig. 10. The
me number of ampere turns will link each of the
ertures in the two ways of winding if the following
lationships are observed:

$$\Delta = \Delta'; \quad K' = \Delta - K; \quad \Omega = \Omega'. \tag{20}$$

Thus the previous analysis is valid for both ways of
nding. In Fig. 2 the drive and output windings are on
fferent legs.

The validity of the analytical considerations in Sec-
on IV has been verified experimentally.

V. Circuit Variation

An increase in gain may be achieved by using a wind-
g through the large aperture of each core that is
orted during the drive pulse applied to the core but
en during the drive pulse applied to the previous
age, causing information input into the core (Fig. 11).
he upper limit on the drive pulse then is removed.
he shorted windings are arranged in series alternately
odd and even stages and normally short all of the
res in the forward direction of the diodes. However,
ring the drive pulse which reads information into the
re, the short is removed by the pulse created by the
rresponding drive pulse. The process requires four

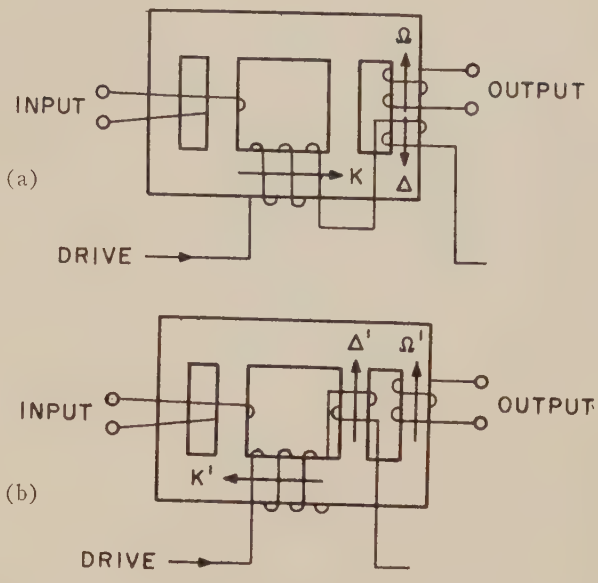


Fig. 10—Two arrangements of windings. (a) drive and output windings both on Leg 4, (b) Drive winding on leg 3, output winding on leg 4, $\Delta = \Delta'$, $K' = \Delta - K$, $\Omega = \Omega'$.

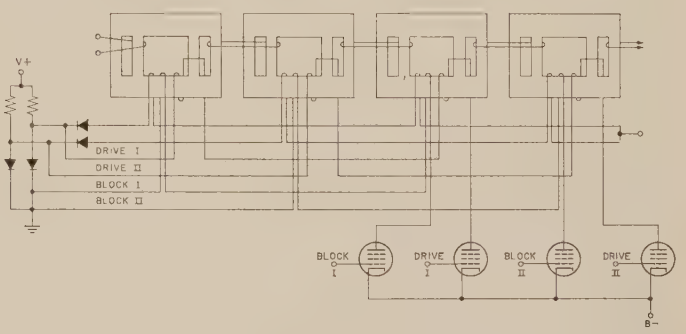


Fig. 11—Increased drive circuit.

additional diodes per shift register, which in themselves do not add much to the component count.

VI. Shift Registers and Delay Elements Made of a Strip Or a Ring of Magnetic Material

Production may be simpler and cheaper if the circuitry is composed of a strip or a ring of magnetic material rather than of individual cores. This is particularly so when utilizing printed circuit techniques. Some circuits of this type are discussed here.

The individual cores of the shift register of Fig. 2 can be laid out as a strip with spacer apertures providing separation between the cores. Such a system is shown in Fig. 12. At the bottom of this figure the flux configuration for transfer of a one is shown. In the system of Fig. 12 the block state in alternative stages is represented by saturation in opposite directions. Comparison with Fig. 2 shows the significance of the difference of the flux pattern.

This type of shift register can be connected in a ring or made out of a ring, connecting the output back to the input.

Fig. 13 shows a scheme which eliminates the necessity for input and output windings between stages. Also,

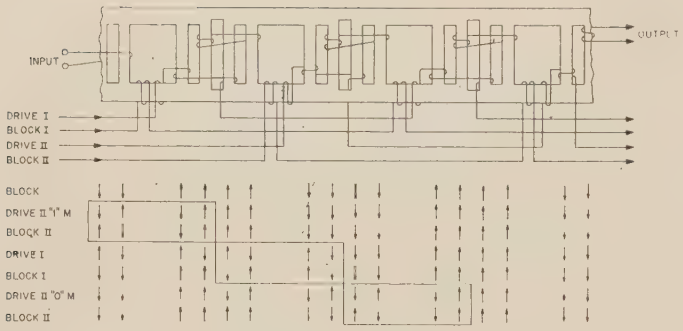


Fig. 12—Arranging the shift register on a strip.

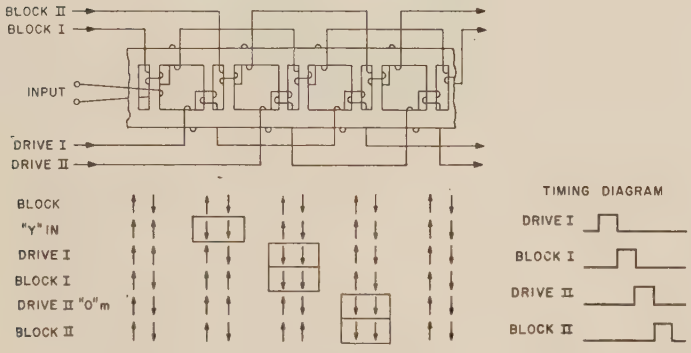


Fig. 13—A strip shift register without input and output windings.

less magnetic material is involved per stage. Even though the losses may be lessened in this way, it is likely that some flux will be lost in the process of shifting information. Since no flux gain can be expected from the turns ratio between output and input windings, the operation may not be stable with information rotated in a ring. However, information may be stepped up through many stages before satisfactory discrimination between ones and zeros is lost.

This type of shift register would be particularly satisfactory then for use as a delay line. Furthermore, for this use only two drive pulses are required. A ring that may serve for this purpose is shown in Fig. 14. A further advantage of such systems is that a bit of information

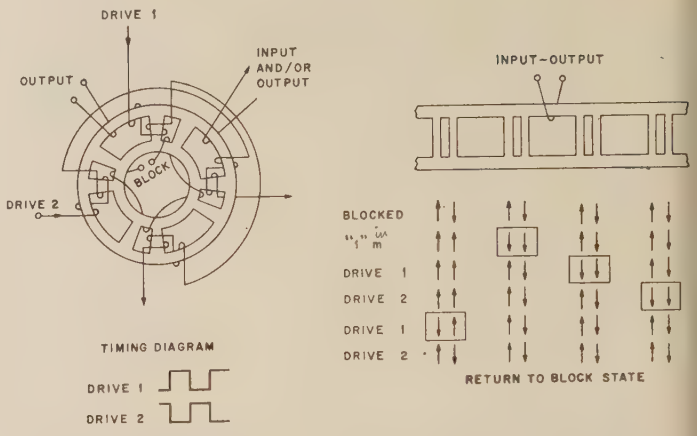


Fig. 14—Delay line utilizing ring of magnetic materials.

transferred corrects the nonregularity in flux pattern remaining in the input stage after turning once around the ring. In doing that, the ring is returned to the blocked stage. This is illustrated at the right of Fig. 14 where the corresponding flux pattern is drawn under the spread ring.

Acknowledgment

The author wishes to thank Nancy E. Remley who has helped in the preparation of this paper.

Bibliography

[1] Wang, A. "Magnetic Delay Line Storage," Proceedings of the IRE, Vol. 39, (April, 1951), pp. 401–407.
[2] Rajchman, J. A. and Lo, A. W. "The Transfluxor," Proceedings of the IRE, Vol. 43 (March, 1956), pp. 321–338.
[3] Rajchman, J. A. and Lo, A. W., "The Transfluxor—A Magnetic Gate with Stored Variable Settings," *RCA Review*, Vol. 16 (June, 1955), pp. 303–311.
[4] Newhouse, V. L. and Prywes, N. S. "High-Speed Shift Registers Using One Core Per Bit," IRE Transactions on Electronic Computers, Vol. EC-5 (September, 1956), pp. 114–120.
[5] Sands, E. A. "The Behavior of Rectangular Hysteresis Loops under Current Pulse Conditions," Proceedings of the IRE, Vol. 40 (October, 1952), pp. 1246–1250.
[6] Fishman, M. "A High-Speed Shift Register Using Magnetic Binaries," presented at 1952 IRE National Convention, New York, N. Y.
[7] Menyunk, N. and Goodenough, J. B. "Magnetic Materials for Digital Computer Components," *Journal of Applied Physics*, Vol. 26 (January, 1955), pp. 8–18.

Correction

M. W. Marcovitz and E. Seif, authors of "Analytical Design of Resistor-Coupled Transistor Logical Circuits," which appeared on pages 109–119 of the June, 1958 issue of these Transactions, have requested that the following corrections be made to their paper.

Eq. (17) on page 113 should be

$$V_{cu1} = \overline{V}_{cu1} \frac{I_{b1}}{I_{b1}} .$$

On the same page, (21) should be preceded by the phrase "From Fig. 5."

The last term of (35) on page 119 should be

$$- \frac{nV_{cc}}{B} \left\{ \frac{\overline{I}_{co1}}{I_{b1}} + 1 \right\} .$$

The squared term in the numerator of the term on the left side of (36) should be l, and not 1.

Correspondence

olding of Symmetric Functions*

Introduction

In this note, $T(X_1, \cdots, X_n)$ is a Boolean nction of n binary variables $X_1, \cdots, X_n$, presenting the transmission between two rminals of a contact switching network. ch a function T is said to be *symmetric on* and X_j provided

$$T(X_1, \cdots, X_{i-1}, X_i, X_{i+1}, \cdots, X_{j-1}, X_j, X_{j+1}, \cdots, X_n) = T(X_1, \cdots, X_{i-1}, X_j, X_{i+1}, \cdots, X_{j-1}, X_i, X_{j+1}, \cdots, X_n).$$

T is said to be symmetric provided T is mmetric on all pairs of variables X_i and in $X_1, \cdots, X_n$. Following the conven- on of Shannon,[1] a symmetric function T ll be denoted by $S_{p_1,p_2,\cdots p_r}(X_1, \cdots, X_n)$ $T=1$ only when any p_i variables are 1 d $n-p_i$ variables are 0, for $i=1, \cdots, r$.

It is well known that a standard contact twork exists to produce any symmetric nction.[1,2] For example, in three variables e circuit is as shown in Fig. 1.

The terminals marked 0, 1, 2, and 3 pro- ce transmissions from terminals a of S_0, S_2, and S_3, respectively. We will say that e row of relay contacts with output termi- l i will be called the ith row, and that the lumn whose output terminal is i will be lled the ith column. Shannon[1] has used e concept of folding for the purpose of mplifying symmetric functions $S_{p_1,p_2,\cdots,p_r}(X_1, \cdots, X_n)$.

Suppose this function can have its cir- it replaced by one in which all contacts ove the p_{s+1} row and on that row can be minated, with the remaining contacts ove the p_s row being connected so that e output of X_j is connected to the junction X_j', X_{j+1}, X_{j+1}', X_j on some row below e p_s row, where $j=p_s+1, \cdots, n$.

If this new circuit has for output $_{p_1,\cdots,p_r}(X_1, \cdots, X_n)$, the original circuit ill be said to *fold down to the p_sth* row.

Furthermore, a circuit for $S_{p_1,p_2,\cdots,p_r}$ $X_1, \cdots, X_m)$ is said to fold completely pro- ded it will fold down to the p_1 row; here is supposed that $p_1<p_2<\cdots<p_r$.

Shannon[3] and Caldwell[4] both remark that folding process will apply to any symmetric nction in which the subscripts are in arith- etic progression. This note gives a neces- ry condition for folding which includes larger class of subscripts.

Theory

heorem 1

If the symmetric function $S_{p,q,s}$ $X_1, \cdots, X_s)$ folds down to the pth row $<q<s)$, then there is an integer m such nat

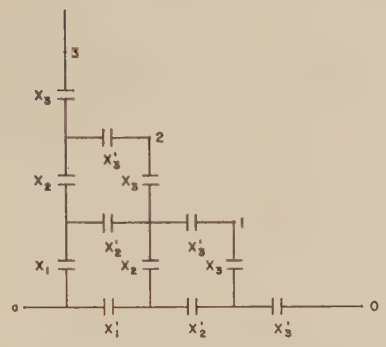


Fig. 1.

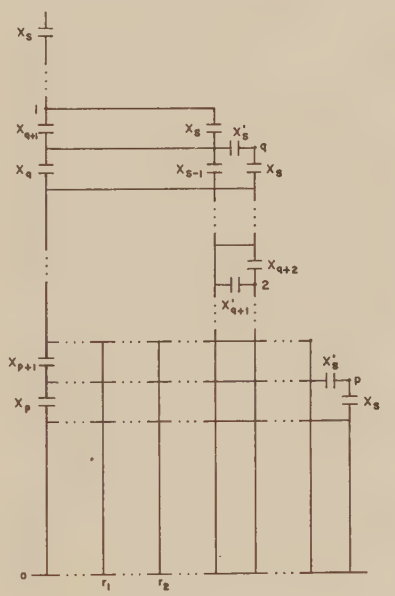


Fig. 2.

$$\frac{s-p}{q-p} = m.$$

Proof: Consider Fig. 2. It is known that none of the relays above the qth row and to the right of the sth column are needed, nor above the pth row and to the right of the qth column.

The first fold will be accomplished by connecting node 1 to node 2. It can be observed that S_s could originally be obtained only by going up the sth column. After folding, $q+1$ open relay contacts are encountered up to the $q+1$ row. From node 2 up to the qth row are $s-(q+1)$ more open contacts, so that S_s is still possible, but now at the qth row terminal, together with S_q.

The second fold will require the output of X_{p+1} on the sth column to be connected to some r_1 column. It will connect just below X_{p+2}. Since there are $q-p$ contacts on the sth column above X_{p+1}, after the first fold, the r_1 column must end at $X_{p+2+[q-(p+1)]} = X_{q+1}$. But then either $q+1=s$ (in which case no folding is possible) or else in the folding process, the r_1 column must have its top contact connected to the r_2 column. In this fashion, since this is a finite system, we can see that the sth column is connected to the r_1 column, that to the r_2 column, $\cdots$, the r_{m-2} to the r_{m-1}, the r_{m-1} to the pth. The number of open contacts on each of the r_i columns $(i=1, \cdots, m-1)$ and on the pth column above the connection from the r_{i-1} row and below the pth row is the same and is equal to $q-(p+1)$. Then for S_s to be obtained at the pth row terminal, there must be a path starting at a on which there are exactly s open contacts. On the sth column there are p below the pth row. Above the pth row on all columns is a total of m open contacts. Hence,

$$s-p = m(q-(p+1))+m$$

or

$$\frac{s-p}{q-p} = m.$$

It can be observed that when $m=2$ the subscripts are in arithmetic progression.

Corollary 1: If $S_{p,q,s}(X_1, \cdots, X_n)$ folds to the pth row (where $p<q<s<n$) then there exists an integer m such that

$$\frac{s-p}{q-p} = m.$$

Proof: S_s has as one minterm the quantity

$$X_1X_2, \cdots, X_sX_{s+1}'X_{s+2}', \cdots, X_n'.$$

This minterm must be realized by the contacts on the nth column. After folding to the pth row, the first $p+1$ open contacts are found on the nth column. Furthermore, that minterm can be realized only by a path up the nth column to the pth row. Then as in Theorem 1, the output of X_{p+1} must be connected to the r_1 row, that to the r_2 row, etc. Again, the part of the minterm $X_{p+2}, \cdots, X_{q+1}$, can be realized only by a path up the r_1 column (since a move to the right would introduce X_b' for $b<s$); and similarly for the r_2, r_3, $\cdots$, r_m columns. The top contact on the r_m column (below the pth row) will be X_s. If it were not, the path $X_1, \cdots, X_{s+1}, \cdots, X_n'$ would be open contrary to hypothesis. Hence, as in Theorem 1, $s-p=m(q-(p+1))+m$ and

$$\frac{s-p}{q-p} = m.$$

Corollary 2: If $S_{p_1,p_2,\cdots,p_r}(X_1, \cdots, X_s)$ where $p_1<p_2<\cdots<p_r\leq s$ folds completely, then there exist integers m_i, $i=0, 1, 2, \cdots, r-3$ so that

$$\frac{p_{r-i}-p_{r-i-2}}{p_{r-i-1}-p_{r-i-2}} = m_i.$$

Proof: It is known that

$$S_{p_1,p_2,\ldots,p_r}(X_1, \cdots, X_s) = S_{p_1,p_2,p_3}(X_1, \cdots, X_s)+S_{p_2p_3,p_4}(X_1, \cdots, X_s) + \cdots, +S_{p_r-2p_{r-1}p_r}(X_1, \cdots, X_s)$$

If the function $S_{p_1,p_2,\ldots,p_r}(X_1, \cdots, X_s)$ folds, then each of the functions in the sum on the right-hand side of the equation must fold. Applying Theorem 1, the result follows.

G. P. Weeg
Computer Lab.
Michigan State Univ.
East Lansing, Mich.

* Received by the PGEC, November 14, 1958.

[1] C. E. Shannon, "A symbolic analysis of relay nd switching circuits," *Trans. AIEE*, vol. 57, pp. 13–723; 1938.

[2] S. H. Caldwell, "Switching Circuits and Logical esign," John Wiley and Sons, Inc., New York, N. Y.; 958.

[3] *Op. cit.*, p. 721

[4] *Op. cit.*, p. 250.

Contributors

Shreeram Abhyankar was born in Ujjain, India, on July 22, 1930. He received the B.Sc. degree in 1951 from the Institute of Science in Bombay, and the Ph.D. degree at Harvard in 1954. During 1955–57, he was Research Associate and Visiting Assistant Professor at Columbia University, and from 1957–58 was Assistant Professor of Mathematics at Cornell University. He is presently Sloan Foundation Fellow and Visiting Assistant Professor of Mathematics at Princeton University. He is a member of the American Mathematical Society.

❖

Robert L. Ashenhurst (M'57) was born in France on August 9, 1929. He was educated at Harvard, where he was a research associate in the Computation Laboratory until receiving the Ph.D. degree in applied mathematics in 1956. In 1956–1957, he was instructor in applied mathematics at Harvard, and is presently assistant professor of applied mathematics in the School of Business and Institute for Computer Research at the University of Chicago. Since 1956 he has been a consultant to Arthur D. Little, Inc.

He is a member of Phi Beta Kappa, Sigma Xi, the Association for Computing Machinery, the Society for Industrial and Applied Mathematics, and the Institute for Management Sciences.

❖

D. D. Aufenkamp, was born in Brock, Neb., on March 31, 1927. He received the B.A. degree from Southwestern at Memphis, Tenn., in 1949, and the Doctorate degree from the University of Paris in 1952.

From 1946 to 1948, he served in the U. S. Navy as an aviation electronics technician. During the summer of 1951, he was a member of the Summer School of Theoretical Physics of the University of Grenoble, Les Houches, France.

Dr. Aufenkamp was an instructor in mathematics at Princeton University from 1952 to 1953, and he taught physics at Southwestern during the summer of 1953. From 1953 to 1956 he was an instructor in mathematics at Reed College, and from 1956 to 1957 was visiting assistant professor of electrical engineering at the University of Illinois. He was on the technical staff of Bell Telephone Laboratories, Inc. during the summers of 1954 and 1955. The following summer he was a member of the applied mathematics staff of Systems Laboratories Corp., Sherman Oaks, Calif., a position he finally maintained on a full-time basis.

He is now with Lockheed Missile Systems Division, Palo Alto, Calif.

❖

Mao-Chao Chen was born in Szechuan, China, in 1926, and received the B.S. degree in electrical engineering from Chiao-Tung University, Shanghai, China, in 1948. After graduation, he worked as an assistant in electrical engineering at Ordnance College, Taipei, Taiwan, China. He came to the United States in the fall of 1955, attending Graduate School in the electrical engineering department of Oklahoma State College, Stillwater, Okla. He received the M.S. degree in electrical engineering from the University of Wisconsin, at Madison, in February, 1957, and then worked for Huggins Laboratories, Menlo Park, Calif.

Mr. Chen is presently studying for his doctorate at Stanford University.

❖

John F. Couleur was born in Chicago, Ill., on July 7, 1925. He is a 1946 graduate of Southern Methodist University, Dallas, Tex., with a Bachelor of Science in electrical engineering. The same year, he joined General Electric's Heavy Military Electronic Equipment Department in Syracuse, N. Y., and for the next several years worked on radar design of the CPS-6B and FPS-6. In 1953, he joined Hughes Aircraft for a year as project engineer on computer system testing. The following year, he returned to General Electric and assumed his present position as project leader on flight test data recording for the ATLAS ICBM radio-inertial guidance system. In this capacity, he is associated with General Electric's Missile Guidance Section in Syracuse, N. Y., which is now producing ATLAS guidance systems.

Mr. Couleur presently has four patents pending, two on binary-to-decimal or decimal-to-binary converters and several on forms of a high resolution discriminator.

❖

Ivan Flores (A'51–M'54–SM'58) was born in New York, N. Y., in 1923. He received the B.A. degree in mathematics from Brooklyn College, Brooklyn, N. Y., in 1948, and the M.A. degree in mathematics from Columbia University, New York, N. Y., in 1949. In 1955 he was awarded the Ph.D. degree in the field of Supervision in Industry from New York University, New York, N.Y.

Dr. Flores has been working in the field of digital computers since 1950. During two years with Mergenthaler Linotype Company, Brooklyn, N. Y., he developed circuits for the automatic electronic justification of typographical materials. He worked for another two years at Balco Laboratories, Newark, N. J., where he supervised research and development in telemetry and automatic control by digital and analog methods. Dr. Flores also conducted investigations in antenna design and physical chemistry at Balco. At the Nuclear Development Corporation of America, White Plains, N. Y., he was engaged in supervising the design and construction of a computer for complete inventory control of the production and stock of the Otis Elevator Company.

At the Remington Rand Laboratories for Advanced Research, South Norwalk, Conn., Dr. Flores has been supervising the electronic development of machines for automatic reading, handling and processing of printed documents. He is now doing the system analysis for a small business computer and is also a part-time instructor in the graduate division of the Polytechnic Institute of Brooklyn, N. Y.

Dr. Flores is chairman of the Membership Committee, IRE Connecticut Section.

❖

Stanley P. Frankel (M'55) was born in Los Angeles, Calif., in 1919. He received the B.A. degree in 1938 and the Ph.D. degree in 1942, both in physics, from the University of California. From 1942 to 1946 he worked for the Manhattan District, U.S.E.D., at the University of California Radiation Laboratory and at Los Alamos. He was on the staff of the Institute of Nuclear Studies of the University of Chicago from 1946 to 1947; a member of Frankel & Nelson, consulting in mathematical physics, from 1947–1949; and on the staff of the Engineering Division of the California Institute of Technology from 1949 to 1954. Since 1954, he has acted as consultant in logical design to a number of firms.

Since 1943 he has made use of a number of digital computers for the solution of various industrial and scientific problems. From 1949 to 1954, he headed the digital computing group which provided for the digital computing needs of CIT. In the course of this activity he developed the logical design of MINAC which was partially constructed in breadboard. This design was subsequently licenced by CIT to Librascope and forms the basis of their LGP-30, the first production copy of which is now in operation at CIT. From 1954–1957, he developed the logical design for CONAC, a more powerful magnetic drum digital computer for scientific use. An engineering prototype of CONAC has been built by the Continental Oil Company and is now in service. More recently, he has designed, in part, a drum computer which is under construction by the General Electric Company.

Dr. Frankel is a member of the American Physical Society, the Association for Computing Machinery, and other professional societies.

❖

E. J. McCluskey, Jr., (S'51–M'55) was born in Brooklyn, N. Y., on October 16, 1929. In 1953, he received the A.B. degree in mathematics and physics from Bowdoin College, Brunswick, Me. Also in 1953, he received the B.S. and M.S. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, from which he received the Sc.D. degree in 1956.

From 1953 to 1955, he was a research assistant and instructor at M.I.T. Since 1955, he has been with the Bell Telephone

aboratories, Whippany, N. J., doing re-arch in connection with electronic switch-g systems. He is a lecturer at the City ollege of New York and at Princeton Uni-ersity, Princeton, N. J.

Mr. McCluskey is a member of Phi Beta appa, Tau Beta Pi, Eta Kappa Nu, Sigma i, and the Association for Computing lachinery.

❖

Nicholas C. Metropolis was born in hicago, Ill., on June 11, 1915. He received he Ph.D. degree in physics at the University Chicago in 1941.

After working on the Manhattan Project the Metallurgical Laboratory at the niversity of Chicago and at Los Alamos uring the war, he was appointed assistant rofessor of physics at the University of hicago in 1946. From 1948 to 1957, he was the Los Alamos Scientific Laboratory, here he was responsible for the develop-ent of MANIAC I and MANIAC II.

Since 1957, he has been at the University Chicago as professor of physics in the De-artment of Physics and Enrico Fermi In-itute for Nuclear Studies and as director the Institute for Computer Research.

❖

Takeo Miura was born in Kyoto, Japan, 1925. He graduated from Kyoto Uni-ersity with the degree of Bachelor of Engi-eering (Kogakushi). He joined Hitachi, td., Tokyo, in 1949, and was engaged in he design of induction motors of Kameido Works. In 1950, he was transferred to Hitachi Central Research Laboratory, where e has since been working on the develop-ent of electronic analog computers, regu-tors, and servomechanisms.

Mr. Miura is a member of the Institute f Electrical Engineers of Japan.

Minoru Nagata was born in Tokyo, Japan, in 1933. He graduated from the University of Tokyo with the degree of Bachelor of Engineering (Kogakushi) in 1956. He then joined Hitachi, Ltd., where he has been engaged in the development of electronic analog computers in the Hitachi Central Research Laboratory. In particular, he has been working on the development of dc amplifiers.

Mr. Nagata is a member of the Institute of Electrical Communication Engineers of Japan, and the Institute of Electrical Engineers of Japan.

❖

Noah S. Prywes (M'55) was born on November 28, 1925. He received the B.Sc. degree in electrical engineering from The Technion, Israel Institute of Technology, Haifa, Israel, in 1949. From 1948 to 1950, he served in the Israeli Navy. In 1951, he received the M.Sc. degree in electrical engineering from Carnegie Institute of Technology, Pittsburgh, Pa., and in 1954, received the Ph.D. degree in applied physics from Harvard University.

Dr. Prywes worked from 1954 to 1956 at RCA, Camden, N. J. as a development engineer engaged in computing systems utilizing magnetics and transistors. From January 1956 to September 1958, he has been a department manager on the LARC project with Remington-Rand UNIVAC, Philadelphia, Pa. In this capacity he was responsible for all high-speed circuit design, interunit transmission, power supplies and control consoles for the LARC system. Later he was responsible for the design, development, construction and test of the computing unit of the LARC system. Since last September, Dr. Prywes has been assistant professor of electrical engineering at the University of Pennsylvania, retaining a consulting position with Remington-Rand UNIVAC in Philadelphia.

He is a member of AIEE and of Sigma Xi.

Frank L. Ragonese (A'46) was born in Bridgeport, Conn., on July 7, 1915. He graduated from the Bridgeport Engineering Institute in 1938 with a certificate in electrical engineering.

After working as an electrical engineer at the Remington Rand Shaver Division in Bridgeport, Mr. Ragonese entered the U. S. Army and served from 1941 to 1946 as an Army Signal Corps Officer. While in the service, he attended radar and ultra-high-frequency courses at Harvard University and Massachusetts Institute of Technology.

Since 1946, Mr. Ragonese has been employed as an electrical engineer by the Remington Rand Laboratory of Advanced Research, South Norwalk, Conn., where he has engaged in the design, development, production and application of both black and white, and color industrial television systems, the design and development of a small magnetic drum unit, special permanent magnet type and electromagnetic type read and write heads for computer use, the development of a helicopter blade dynamic tracking system and, most recently, in the development of a magnetic character reading system.

❖

N. Rouche (M'58) was born in Huy, Belgium, on June 2, 1925. He was graduated Ing. Elec. from the University of Liège, Belgium, in 1950.

After a year as lecturer on the Faculty of Sciences, University of Liège, he did research on nonlinear oscillations at the Institute for Mathematics and Mechanics, New York University, New York, N. Y.

During the years 1953–1957, Mr. Rouche was an engineer in the Department of Automation at the Bell Telephone Manufacturing Company, Antwerp, Belgium. In 1955, he was graduated Agrégé de l'Enseignement Supérieur from the University of Liège.

Mr. Rouche has been professor of applied mathematics and electronics at Lovanium University, Leopoldville, Belgian Congo, Africa, since 1957.

PGEC News

Call for Papers

The 1959 Solid-State Circuits Conference, sponsored by the IRE Professional Group on Circuit Theory, AIEE Committee on Electronics, IRE and AIEE Philadelphia Sections, and the University of Pennsylvania, will be held on February 12 and 13, 1959, in Philadelphia, Pa. Plans for the 1959 Conference call for the presentation of papers dealing with circuit properties, circuit philosophy, and techniques related to solid-state devices in the following general areas:

Novel semiconductor devices and conventional semiconductor devices in novel modes of operation.
Significant contributions to the transistor circuit art.
Solid-state devices performing an integrated circuit function.
Solid-state microwave devices and amplifying mechanisms.
Solid-state memory, storage, and logic devices.
Low-temperature digital and linear applications.
Novel types of solid-state devices.

For further information, write to: Arthur P. Stern, Program Committee Chairman, 1959 Solid-State Circuits Conference, General Electric Company, Electronics Park, Building 3—Room 113, Syracuse, N. Y.

JOINT COMPUTER COMMITTEE

SENEWS

SCIENCE EDUCATION SUBCOMMITTEE NEWSLETTER

Vol. 1, No. 3 December, 1958

Purpose

It seems worthwhile to reiterate the purpose of *SENEWS* and to stress the need for your contributions if the program is to succeed. *SENEWS* is a newsletter addressed primarily to computer-oriented members of the IRE, ACM, and AIEE to aid them in promoting interest and knowledge in computing among high-school age students. It wants also to communicate directly with secondary school teachers and other science-education groups, to report on their projects and problems and offer help in bringing together all those interested in such projects.

The response to date has been quite gratifying, with the majority of items coming directly from teachers or leaders of other groups. We do, however, need and continue to solicit your contributions and comments.

Write to:

Michael Warshaw, *Chairman*
JCC Science Education Subcommittee
The RAND Corporation
1700 Main Street
Santa Monica, Calif.

Changing of the Guard

With the publication of this issue, we announce a new JCC Science Education Subcommittee Chairman and a new AIEE representative to the Editorial Board.

Replacing Clare Farr as Chairman and Editor-in-Chief of *SENEWS* will be Michael Warshaw of The RAND Corporation. Robert Goldman of Bendix Computer Division is taking over from G. N. Hollander as the AIEE Editorial Board Member.

As before, R. W. Melville of Stanford Research Institute and G. E. Forsythe of Stanford University will remain as representatives of the IRE and ACM, respectively.

A vote of thanks to Messrs. Farr and Hollander for their efforts in getting *SENEWS* through its nascent phase.

Distribution and Preprints

Volume 1, Nos. 1 and 2 of *SENEWS* were carried in the *Communications of the ACM* and IRE Transactions on Electronic Computers. We are pleased to announce that the AIEE is going to distribute *SENEWS* separately to interested members and will make preprints available of No. 2 and all succeeding issues at their standard charge. Interested parties may obtain these by writing:

R. S. Gardener
American Institute of Electrical Engineers
33 West Thirty-Ninth Street
New York 18, N. Y.

Future Engineers of America—Philadelphia

Daniel Ashler, from the Math Department of Abraham Lincoln High School in Philadelphia, has reported on the activities of the group he sponsors: two decimal-to-binary converters have been built, and an adder and multiplier are nearly finished. Ultimately, some sort of computer may be assembled.

The designs for the decimal-to-binary converters are original: the two solutions to the problem are quite different, and one of the boys has devised a circuit notation for which he claims superiority over the conventional schematic diagrams. One of the converters won an outstanding award from the Ford Motor Company's Industrial Arts Award competition. Components were gleaned from worn out pinball machines contributed by a local dealer and from scrap piles in junk yards except for a few switches that had to be bought at a "war surplus" store.

The boys are studying Boolean algebra and symbolic logic, and they have been taught to code for the IBM 650. Last spring they worked out programs involving three-stage looping and ran them on the 650 at the local IBM office.

Time has not permitted us to get all the details on the conversion schemes or on the new notation. They

hould both prove to be interesting, and we hope to ave more information for the next issue.

On the editorial side, we would like to commend Mr. shler and his boys for showing what we feel is remarkble restraint in not plunging right in with a full-lown computer project, but rather taking it piece by iece until a sufficient backlog of experience has been ained. Projects such as the design and construction of converter or an adder are fascinating tasks in themelves and have the added feature of reaching compleion in a finite length of time.

Future Engineers of America—Van Nuys

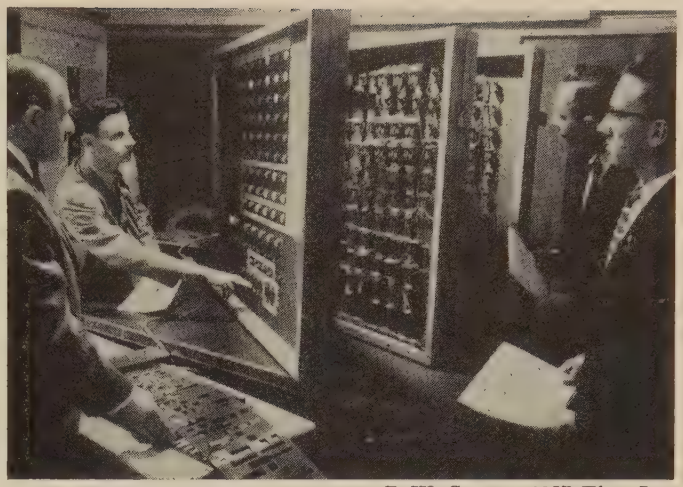
The FEA in Van Nuys, Calif., has recently started a omputer laboratory where each year ten students will eceive theoretical and practical training in digital comuting. The technical training will be supplied by Don nderson and Jim Russell from the local IBM office. wo evenings a week they will give instruction in comuter logic, circuit theory, and programming. Sufficient omponents will also be made available to give the tudents realistic laboratory experience. It is planned o make this a permanent program, accepting ten new tudents each year, and Gerald Speen of the Valley ouncil FEA estimates that a fully operating computer ay be realized in five years.

There is, however, still a great need for professional elp at this section of the FEA. According to Mr. Speen, hirty-five students want to work in some phase of omputing, but only ten could be selected. So any comuter engineers who live in the valley and would like o spend one night a week helping out a few interested igh-school students should contact:

Gerald Speen
Future Engineers of America
Box 2664
Sepulveda, Calif.
Phone—EM 72211

High-School Computer

Aaron Buchman and his mathematics students at Hutchinson Central Technical High School, Buffalo, N. Y., are shown in the accompanying photograph with he relay computer they designed and built (see *SENEWS* Vol. 1, No. 1). The relays are arranged on hinged anels so that the student can actually see each step f the computation as it is performed by the machine. Note the punched "paper" tape (9-inch wide pressoard) which the student feeds into the tape reader by and. In this manner, he controls the speed of computation or stops it at any desired point in the program. The conceptual advantages of this computer for elementary instruction can hardly be overstated. Mr. Buchman staged a most convincing demonstration for our *SENEWS* editor last August, and is extremely nergetic and cooperative in personally answering inquiries from others who wish to build a school computer.



F. W. Goro, © 1957 Time Inc.

Palo Alto Community Science Seminar

Although the following report from George Forsythe is not devoted primarily to the computing art, we feel it is of sufficient general interest and importance to print without deletion.

"The Joe Berg Foundation (1712 South Michigan Ave., Chicago 16, Illinois) is described in the *Reader's Digest* of August, 1958. It is subsidized to help communities set up and run their own science seminars for gifted high school students with a real interest in and talent for scientific study. Things worked in Palo Alto something like this: at the initiative of someone, probably Henry Martin, science teacher at Palo Alto High School, the school district contacted the Joe Berg Foundation. The foundation sent a number of standard tests and advice on how to use them to select appropriate students. In the spring of 1958, certain students of the 9th, 10th, and 11th grades, recommended by their science or mathematics teachers, took the test. Of these students, 35 were selected. Meanwhile, Martin rounded up a comparable number of scientists in the area from industry and university circles.

"One night Jacob Shapiro of the Berg Foundation met with the scientists, the students, and their parents. He explained the general nature of the program, the need for regular meetings, and so on. He envisioned individual project work between youth and scientist, plus general lectures and outside reading. It was made clear that the Berg Foundation has no control over the program, which is run by the group itself.

"Since last April the group has met from 7:30 to 10 P.M. each Wednesday evening at Palo Alto High School, with no break for vacation. From 7:30 to about 8:30 there is an expository lecture, usually by an adult scientist or engineer. After questions and discussion, the group breaks up into sections from 9:00 to 10:00. Sections have been formed for biology, physics, chemistry, mathematics, electronics, geology, astronomy and possibly others.

"Here are some of the titles of the general talks: 'Radiological Safety Procedure,' 'The Electromagnetic

Structure of the Proton and the Neutron,' 'Deformation of the Earth's Crust,' 'Immunity,' 'Genetics,' 'Instrumentation Chemical Analysis,' 'Nuclear Reactors,' 'Microwave Tubes,' 'Magnetic Resonance,' 'Problems in Mathematics,' etc. The speakers have included at least one Nobel prize winner, and numerous persons with an international reputation.

"For the most part, the activities of the groups consist of individual projects worked on by one student and one adult—often in the adult's laboratory. The following projects are typical of those proposed: study of variable stars, study of the effects of a nonlinear circuit element, study of an optimal automobile turn indicator, fluid flow visualization, a magnetic survey, and seasonal variation in quality of the Palo Alto water supply.

"In the mathematics section we have so far emphasized group study more than individual research, and are alternating two activities. Benjamin Epstein is leading the group in statistics using the excellent text by the College Entrance Examination Board, 'Introductory Probability and Statistical Inference for Secondary Schools.' George Forsythe is leading the group in learning to code and operate the IBM computer type 650. The IBM manual is the text, to be supplemented, when it arrives, by R. V. Andree's 'Programming the IBM 650 Magnetic Drum Computer and Data Processing Machine' (Holt, 1958). At the moment, the group is coding the Monte Carlo evaluation of the area of a circle, combining coding and experimental statistics. Computer engineers will be interested to know that Bill Kautz of the Stanford Research Institute discussed logical circuits and demonstrated his electronic computer components.

"We are all learning how much bright youngsters can do, given some intellectual leadership by mature scientists. And the students are learning that membership in the group gives them real prestige in their schools."

University of Oklahoma Teacher-Student Computer Course

Because of the enthusiastic response given the 1957 University of Oklahoma's lecture series in modern mathematics for high school students and teachers, a full semester, noncredit course in computer programming and an introduction to related mathematics has been started for the coming school year. The history behind this program has been communicated to us by Dr. Richard Andree from the University of Oklahoma.

In September, a one-page announcement of the class was sent to high schools in the area, stating the goals and establishing October 4th as the starting date. The requirements were that no teacher would be admitted without an accompanying student and conversely. The announcement further made it clear that a great deal of homework would be required, the course would be free of charge, but the student would have to furnish his own transportation and text. Not everybody could be accepted. Classes would meet for a full day on alternate Saturdays, with the mornings used for laboratory work and the afternoons for lectures.

Because of the competing activities at school, such as service teacher training, football, scheduled trips, etc., plans were made to accept no more than 30 students. Almost 300 applications were received! Hurried consultations and changes in plans allowed the program to be enlarged first to 90 and finally to 106 persons, as it was difficult to turn down students who were willing to commute 300 miles one way to take the course! The teachers were chosen first, then each teacher was permitted to select one of his students.

A grant from the Oklahoma Frontiers of Science Foundation is providing the reimbursement for the instructor and two laboratory assistants, and the University of Oklahoma furnishes classroom space and machine time. The text is R. Andree's "Programming the IBM 650" (Henry Holt Co.).

Dr. Andree reports that enthusiasm is running high, and the resulting programs are well written and often ingenious. He said that it is proving to be a rich, heartwarming experience to teach a group like this, and he sincerely recommends it to other computer installations.

Index to

IRE TRANSACTIONS

ON

ELECTRONIC COMPUTERS

Volume EC-7, 1958

IRE Transactions on Electronic Computors

Index to Volume EC-7, 1958

Contents

Index to Authors

(Listed by month and page.)

Index to Subjects

(Listed by month and page.)